



Priyaranjan Singh

Mobile Multi-Camera Remixing Client

Helsinki Metropolia University of Applied Sciences
Master of Engineering
Degree Programme in Information Technology
03 May 2012

Acknowledgements

This thesis is the result of the work that I have been carrying out at Nokia Research Center in the challenging area of mobile content sharing. The work was carried out during 12 months.

I would like to express my gratitude to the supervisors, Dr. Igor Curcio and Mr. Sujeet Mate from Nokia Research Center and examiner Dr. Mahbub Rahman at Helsinki Metropolia University of Applied Sciences for their technical support, guidance and comments. I thank senior lecturer Taru Sotavalta for being my language instructor. I would also like to thank my family for their support and encouragement to make this thesis happen.

Tampere, May, 2012.

Priyaranjan Singh

Author(s)	Priyaranjan Singh
Title	Mobile Multi-Camera Remixing Client
Number of Pages	76
Date	03 May 2012
Degree	Master of Engineering
Degree Programme	Information Technology
Specialisation option	Mobile Computing
Instructor(s)	Dr. Mahbub Rahman, Professor & Sr. Technology Expert Dr. Igor Curcio, Research Leader Mr. Sujeet Mate, Sr. Researcher
<p>The main purpose was to study nature and requirements of event participants in order to provide a practical and feasible method for sharing the content that is captured on the mobile phones. The goal of this project was to develop a client application to analyze conditions best suited to use mobile devices for uploading and conditions suitable to use PC for upload.</p> <p>The Qt SDK was used to develop a content sharing mobile application to organize images and videos based on recorded events. This client application was expected to provide an interface to video editing service to browse, preview and upload media of one or more event(s). A PC based application was also needed to synchronize videos/images from mobile client and upload to web service. The Mobile application was tested on a Nokia N8 mobile phone.</p> <p>The thesis presented a solution for content-based event sharing and proposed a user friendly approach to upload content from mobile devices. It encourages end-user participation in social media services, by enabling content upload at minimal latency. Our study suggested running video editing/merging logic at central location server to combine all the uploaded video and images into one master copy called remixed video.</p> <p>The prototyped client application was found important for content-based event sharing approach. The client application provided an interface to web based online remixing service which utilizes multiple videos to generate the video remixes. The client application was also used to test few media uploading methods.</p>	
Keywords	Mobile Phones, Video Remix, Event Based Upload, Qt, QML

Abbreviations

ADSL	Asymmetric Digital Subscriber Line
AP	Access Point
AVR	Automatic Video Remix
EBUTM	Event-based Uploading Technique for Mobile Devices
FAT	File Allocation Table
FTP	File Transfer Protocol
GB	Giga Bytes
HD	High Definition
HTTP	Hyper Text Transfer Protocol
IMEI	International Mobile Equipment Identity
MC	Mobile Client
MC-PCMw	Mobile Client-PC Based Middleware
MD	Mobile Device
MMCRC	Mobile Multi Camera Remixing Client
PC	Personal Computer
PCMw	PC-based Middleware
WEM	Web-based Event Manager
WLAN	Wireless Local Area Network

Contents

Acknowledgements

Abstract

Abbreviations

1	Introduction	7
2	Background	11
2.1	Event	11
2.2	Event Management	11
2.3	Overview of Event-based Uploading Technique for Mobile Devices	15
2.4	Related Art	17
2.5	Limitations of Mobile Upload Techniques	19
3	Architecture	20
3.1	Building Blocks	21
3.1.1.	Mobile Multi Camera Remixing Client	21
3.1.2.	PC-based Middleware	24
3.1.3.	Web-based Event Manager	26
3.2	Communication Protocol	26
3.2.1.	Hypertext Transfer Protocol	26
3.2.2.	Universal Serial Bus	27
4	Reference Implementation	29
4.1	Mobile Multi Camera Remixing Client	29
4.1.1.	Setup for Hardware and Software	30
4.1.2.	Design	31
4.2	PC-based Middleware	51
4.2.1.	Setup for Hardware and Software	51
4.2.2.	Design	53
5	Upload Performance Measurement	59
5.1	Measurement Setup	59
5.2	Test Files	60
5.3	Test Results	61

5.4 Result Analysis	63
6 User Feedback.....	65
7 Conclusions.....	71
References	73

1 Introduction

In today's world with the advances of mobile technologies and social networking mobile devices are not just phones but becoming mobile entertainment, mobile television, or mobile commerce. This study brings mobile multimedia, social events and content upload together for one purpose, which is mobile multimedia content sharing for events.

Mobile multimedia refers to the multimedia information exchange over wireless networks or the wireless Internet. The popularity and evolution of mobile computing devices, together with fast, affordable mobile networks, have made it possible to increase the range and complexity of mobile multimedia applications and services provided to end-users of these kinds of equipment. In the core of this revolution is the astonishing growth in the number, types, novelty, and complexity of mobile multimedia applications and services [1]. Mobile multimedia has become common with the ubiquitous availability of the camera, ranging from low-resolution to high resolution videos and pictures depending on mobile device capability.

Social events are a gathering of people for the purposes of socializing, conversation, or recreation [2].

In the era of the 21st century, social networking is an easy way to interact with people. Now 35 percent of adults on the Internet have a profile on at least one social networking site, and 51 percent have more than one and three-quarters of users between the ages of 18 and 24 have an online profile. The Pew Research Center found that 89 percent of these people use the sites to keep up with friends, 57 percent to make plans with friends and 49 percent to make new friends [3].

When people are spending more than half of their time online with multimedia content gives an idea of how important social networking is today. Multimedia content is acting as the fuel and mobile devices as the tool for social networking. So there is a need for being able to share all the information that a person would like to share. Here comes

the need for social networking applications. These applications are useful, entertaining and addictive in nature.

Content sharing is one of the most successful trends in society. At public events, a significant number of people carry mobile devices capable of capturing video, detecting the device location and orientation in space, connecting to the Internet.

Now the user is more and more willing to find content sharing applications. Few years ago an email client was the favorite and trusted tool for content sharing but now the trend is changing. This is happening after the rise of Facebook and YouTube. Every big market player in software industry is now trying to make their contribution due to rise in public demand and to fulfill people's taste. People's taste is not static, which makes this area very demanding and dynamic [4].

Mobile phones with an integrated camera have a big social impact. Photography and videography allows people to capture personal and group memories and also maintain social relationships. Mobile phones can now record HD videos and editing of the recorded videos has become a useful feature, since users find it convenient to refine or customize video content on the same device it was captured on. However, video editing requires a lot of system resources and mobile phones have quite limited computational resources, memory and battery power. This results in need of online video editing services that can perform techniques that are not as resource constrained as those on mobile devices. Another variant of online video editing is the remixing which utilizes multiple videos to generate the video remixes. Consequently, there is a need to transfer or upload the high quality content (videos and pictures) that is captured using the mobile phones to online video editing and remixing services. High quality and high resolution content requires a large amount of data transfer. This creates a need to investigate and implement a practical and feasible method for sharing the content that is captured on the mobile phones, with an online remixing service.

User requirement of the thesis starts from the idea of social event attended and recorded by various people. Later, the attendees perform separate uploads to social networking channel like YouTube, Flickr or any suitable place. Some of them edit

videos; others leave the recorded videos or pictures as it is, some of the attendees' employ professional equipment while others use mobile phone cameras for recording the content. Videos taken by all of them in a way are different in terms of quality and views, due to the difference in their recording equipment and their positions in the event. However all of them have a challenge of transferring the large video and picture files to the web service.

A goal of this project is to understand the nature and requirements of event participants in order to provide suggestions for the best possible event sharing mechanism. A goal is to analyze conditions best suited to use mobile devices for uploading and conditions suitable to use PC for upload.

The following list includes points useful for designing content sharing software and uploading tips for event participants. On the basis of these factors, the user decides on whether to use a mobile or PC as the uploading device.

1. The experience of capturing videos and pictures at the event is fresh right after the event and becomes less clear with time. Thus it is more beneficial for the user and the service to make use of the content as early as possible.
2. Variability in the amount of content captured.
3. Availability of network connectivity and bandwidth
4. Battery power consumption
5. Cost of data uploads
6. User movement
7. CPU usage for upload affecting other application performance

In order to achieve goal of the project, a content sharing client application is designed and developed to provide a user-friendly mechanism for contributing large amount of content to various social network services. On the basis of above factors client application uses situation-based uploading approach which will be discussed in more detail in the coming chapters.

In Chapter 2, the background of the thesis, uploading techniques prior arts references and advantages are discussed. Chapter 3 introduces the proposed architecture of

event-based uploading techniques and explains the reasons behind the design decisions. It discusses the features of uploading techniques and their problems. Chapter 4 presents the setup needed for the reference implementation and its detailed design. It also presents the details about client application functionality as well as its look and feel. Chapters 5 do analysis of various uploading techniques and suggest the best one. Chapter 6 presents the user feedback for concept and the client application. Chapter 7 concludes with the finding achieved during study and about drawbacks.

2 Background

This chapter describes the event-based uploading technique at a high level and describes some details of important design issues. Special attention is paid to how to implement Event-based Upload Technique for Mobile devices that is EBUTM. Motivation and reasoning for EBUTM are also presented.

2.1 Event

In the context of this thesis, an event is a social gathering where users record videos and pictures [\[2\]](#).

2.2 Event Management

Event management is the way of handling programs that are either business, social or combination of two. Event management handles events for a small or large number of users. Planning event is a time consuming and stressful affair which requires special expertise. One of the important aspects of a successful event is event recording. For a reliable and effective program it is good to have one event management team. Size of the team can grow according to the program size [\[5\]](#).

In the scope of this thesis, all the event participants sharing their videos and pictures from the event are considered as contributors. An event is considered a unique occasion and the participants are encouraged to share their media files in a common location. The event management function generates remixes of the event and provides access for the remix to the participants. The reference implementation of the thesis worked on the idea of below mentioned event management use case in Figure 1.

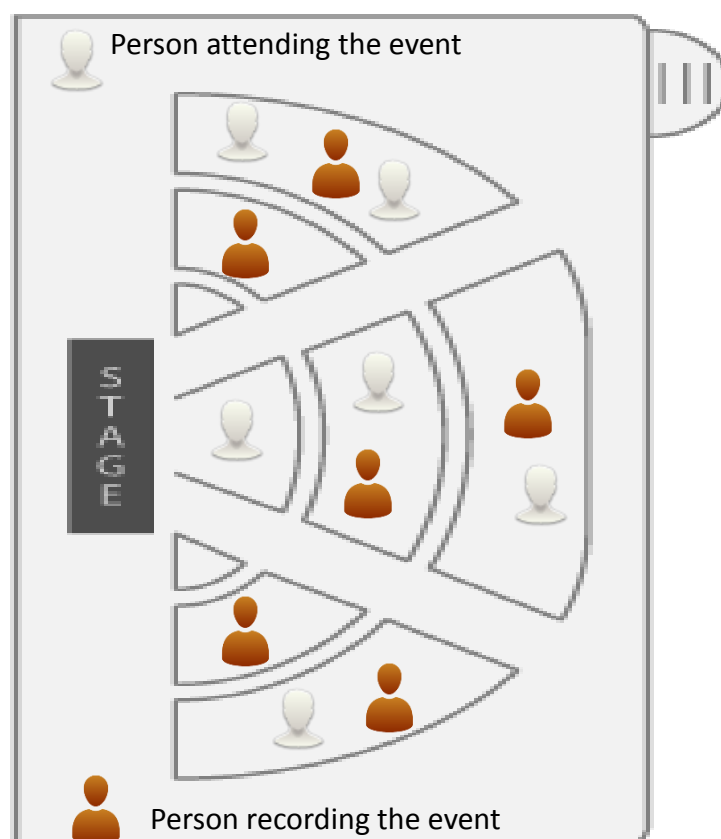


Figure 1: Stage for Event Recording

In Figure 1 the event is a stage show, which is attended by many people some of whom are event contributors (a subset of persons recording videos and pictures in the event).

The conventional approach for video recording is such that every individual records in the event and shares/uploads media files to a social networking forum separately and independently. The same event instance generated on social media websites multiple times and by multiple people. This takes away the unique identification of the event.

Better approach is, for each event one of the contributors creates an "event" entry at the website and the subsequent users contributing to the event can upload the videos and pictures captured by them. Event management host remixes received media files based on analyzing the received data from the users. This Automatic Video Remix (AVR) of the event recording contains the best parts of the contributed content from the event. The resultant remix video tells the detailed story of the event as it gives the wide event coverage which is virtually not possible to be achieved by a single user.

As illustrated in Figure 2, this thesis aims to facilitate maintaining a unique event by enabling multiple users contribute their captured media.

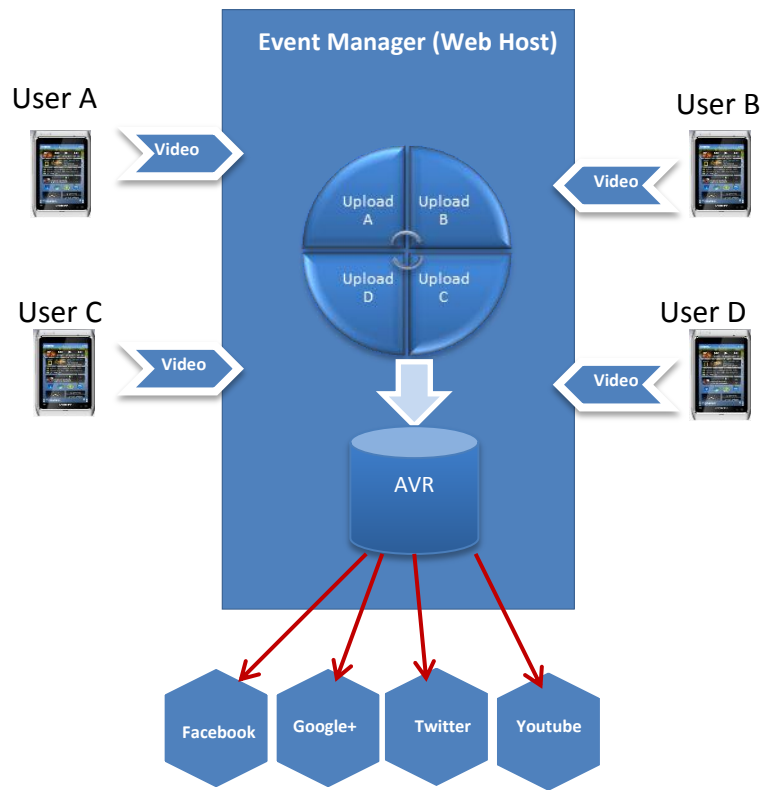


Figure 2: Event Management

Note that AVR is a smart algorithm which works by remixing uploaded videos recorded from various angles, positions and time instances. It also optimizes the combined video as uploaded videos from multiple users may differ in terms of video camera and the photographer's skill level as well as position.

In general, media management related to an event can be summarized in 4 steps. These steps are mentioned in Figure 3.

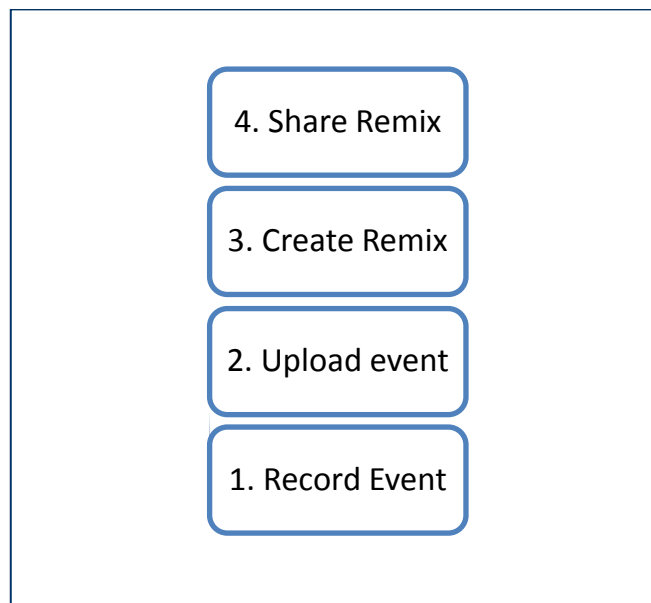


Figure 3: Event Media Management

As shown in Figure 3, event recording is the first step where various people attending the event record videos and take pictures. In the second step, people contribute by uploading their videos and pictures. Reference implementation of the thesis works on this step, and provides the contributor an easy solution for uploading images/videos. To provide an easy solution this thesis defines the EBUTM technique, which is explained in the chapter 2.3. In the third step of event media management, the remote event manager collects the uploaded video and creates an AVR. In the fourth and last step, the AVR video is made available to the event attendants/contributors and later the AVR video may be uploaded to the users' favorite content sharing channel such as Facebook or Google Plus.

2.3 Overview of Event-based Uploading Technique for Mobile Devices

Event management system uses event-based uploading technique (in short EBUTM) to upload media files to the web service. EBUTM uses two uploading approaches as shown in Figure 4. During study of content sharing and EBUTM technique software development is done for client application for both, a PC and a mobile.

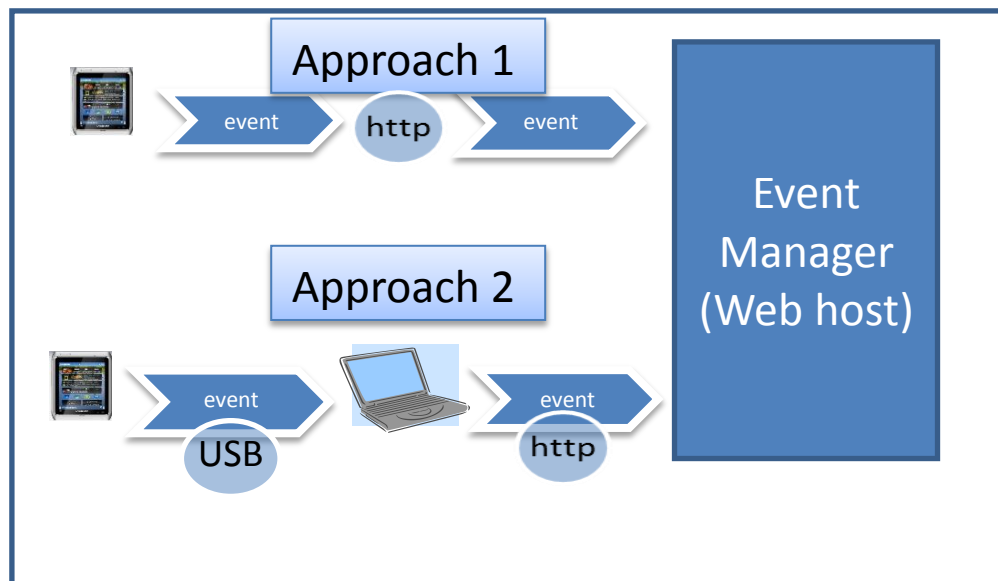


Figure 4: EBUTM technique

1. **Uploading via Mobile Client (Approach 1 or A1):** While uploading media directly from the Mobile Client (MC), the media recording as well as the upload occurs from the mobile device, for example uploading via a mobile phone using WLAN or cellular network.
2. **Uploading via PC (Approach 2 or A2):** While uploading media via a PC, media shooting happens from the mobile device whereas the media upload happens from a PC. In this document, the word PC is used in a more generic manner, which could mean a suitable device that is not power-constrained and is connected to Internet such as Tablet or laptop.

EBUTM technique tries to utilize the merits of both the approaches mentioned above and minimize their respective demerits. The optimal media upload solution varies

depending on different upload situations and the users' requirements. Figure 5 shows a few of the use cases for upload technique selection.

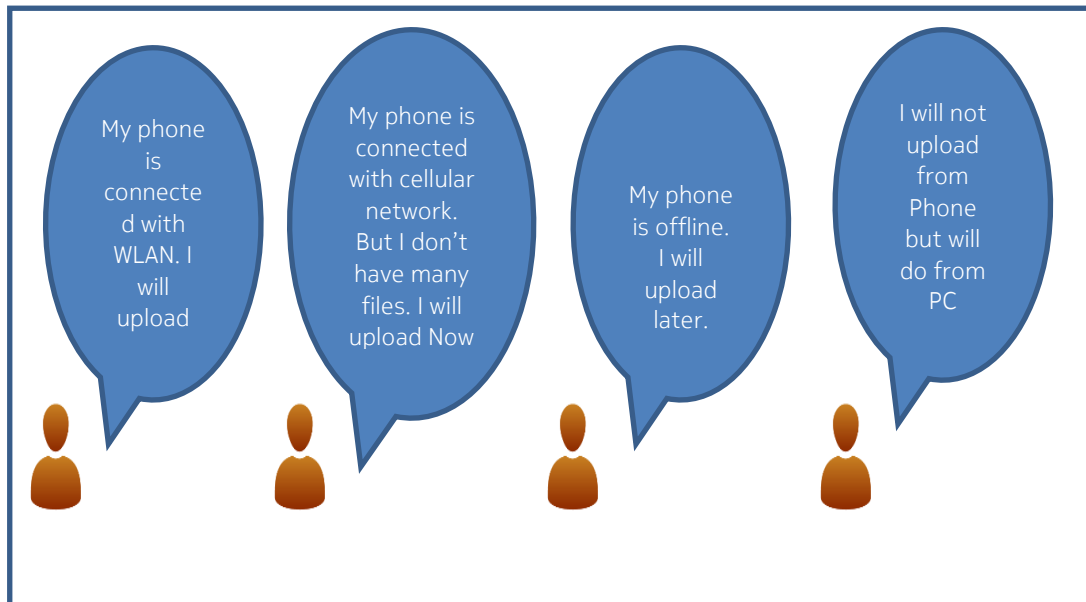


Figure 5: EBUTM upload selection choices

The following three points need to be taken into account in deciding whether approach A1 is more suitable or A2.

Device configuration: If the uploading device is low on battery or processing power constrained, then approach A2 may be more suitable for uploading large sized media data. Extra CPU usage during upload also affects performance of other running application. However if MC constantly getting power supplies, then A1 can also be used.

Event Duration: If the event under consideration is a long-duration event, there is a higher chance of recording large amounts of content. A2 would be a more suitable approach if the amount of data to be uploaded is very large. Approach A1 can be used when event duration is on the lower side. One of the deciding criteria is that a user may not like mobile devices to be always busy uploading media during the event.

Network: If a mobile is connected with the WLAN, the upload is faster and cheaper. The uploading time will be nearly the same compare to PC upload. Hence A1 approach

can be used considering event duration. An upload using a cellular network may be slower and more expensive depending on the type of the network and the billing arrangement. The speed of the upload also varies depending on the network type that is 2G, 2.5G, 3G, 3.5G and LTE. As the network upload speed increases, the media upload is faster. A2 approach can be used considering issues in cellular network quality as well as cost. In the case of WLAN, the A2 approach may be preferable if the user is not likely to stay in the vicinity of the available WLAN AP during the expected upload duration.

Uploading and remixing (editing) recorded media is not a new technology; couple of related art can be found from internet. Chapter 2.4 mentions some prior arts close to the study done in the thesis.

2.4 Related Art

Video recording, editing and uploading using a mobile phone cannot be called a new technique; there are numerous types of mobile phones with different levels of an integrated camera, capable of editing videos and also capable of uploading videos in the market. They use special algorithms for video compression to reduce storage space required on the device as well as reduce the bandwidth needed for transmission over the network. There have been huge improvements in the processing power of the mobile chip-sets that are now capable of performing complex computations. Network capacity for data communication has also improved. Various online video editing tools are available for a variety of video formats.

Kaltura online video editor is remixing service, but this service does not offer a mobile based upload mechanism [6].

Other online media portals such as Flickr offers either mobile-based upload option or PC-based upload, but does not offer a solution that combines the higher accessibility of mobile devices and unconstrained battery power of PCs.

Paper about video remixing: “We Want More: Human-Computer Collaboration in Mobile Social Video Remixing of Music Concerts” talks about video remixing [7] which refers to a similar application scenario which is the subject of this thesis.

These are few patent applications about uploading techniques:

Geo-predictive real-time media delivery in mobile environment:

The prior art proposes a way to improve client media quality in Geo-Predictive Media Streaming or other media streaming applications based on location information. This prior art is about predicting network delays and outages based on location information and smoothening the video streaming and playback experience [8].

Method and apparatus for providing a Geo-predictive streaming service:

The prior art proposes to leverage collaboration between users (crowd-sourcing) to detect temporary outages along a path and using active geo-probing (e.g., locating areas of congestion). This prior art also about ensuring smooth playback using geo-location [9].

Predictive bit-rate modification of content delivery in a wireless network:

A sender in a wireless network may adjust the encoding bit rate of the transmitted content and/or the transmission bit rate of the content based on the predicted future channel throughput at a predicted future geographical position of a client mobile device, such as a cellular telephone. By appropriately adjusting the bit rate prior to the client mobile device experiencing the predicted low throughput, the likelihood of continuous content consumption by the client mobile device, even while experiencing the predicted low throughput conditions, may be increased. The prediction may be performed at the network side and/or at the client mobile device side.

This prior art is about adapting the encoding bitrate based on expected change in network throughput whereas this prior art is related to scheduling upload of selected content [10].

2.5 Limitations of Mobile Upload Techniques

Even many features have improved in mobile phones; the memory is a constraint to this day. For example, compressed 720p HD videos and audio take approximately 80 MB size for one minute of recording. Also there is a file size limit of 4 GB owing to the FAT32 file system if it is used in some mobile devices.

It is still a difficult task to upload multi-gigabyte sized files from mobile devices considering battery life and processing power. Also, the available wireless network connectivity and bandwidth are constraints, which are essential factors that are external to the mobile device. Many of the external storage devices such as memory cards use FAT32 which suffer from above mentioned file size limitation [11].

Success stories about media-sharing have raised the question of how to elevate to the next level in the mobile markets. This is possible by recognizing the issues listed below, without which, a full-fledged solution satisfying all the needs is not possible [12].

- Memory constraints
- Battery constraints
- Bandwidth-limitations in cellular network
- Insufficient bandwidth leading to large upload times
- Lack of network connectivity
- Error prone networks
- Cost of data upload
- Variable signal strength

The above factors mean it is always not advisable to immediately start uploading the files. In addition to available network characteristics, other factors like user's movement, battery consumption pattern, the device upload behavior. The other side of the constraints is the application requirements which influence the latency requirements for upload. The multitudes of factors mean the decision whether to start the upload immediately or defer it for later is complex.

3 Architecture

The chapter describes the overall architecture of the EBUTM and presents information about the event management related functionality that is included in the system without going into the details of the system operation. EBUTM uses client server architecture where data flows between application nodes that is Mobile client (MC) and up to two service nodes a PC and/or a web server.

Mobile device's camera application does the image and video recording, PC act as a gateway which takes the videos and pictures to be uploaded from the mobile device and then uploads it to the web server (or service) that is accepting and hosting the uploaded media files. The web service accepts the videos and pictures from various users and processes the data for the specified events.

The EBUTM architecture design moves around its both approaches A1 and A2. As shown in Figure 6 functional architecture of A1 and A2 uses predefined logical blocks.

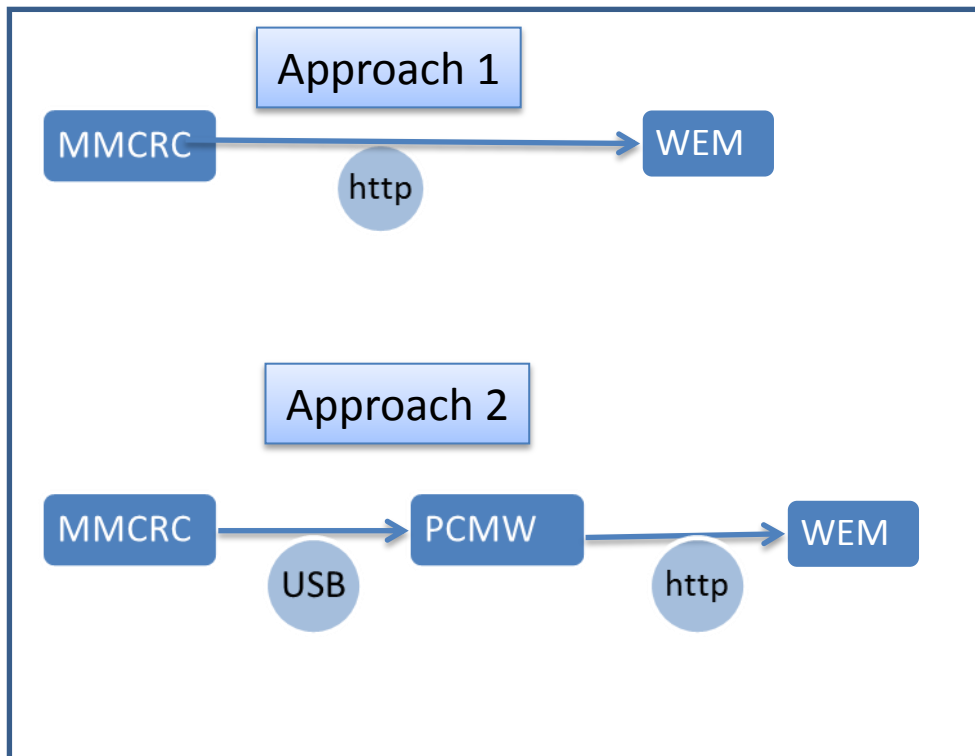


Figure 6: EBUTM Uploading Interfaces

As shown in Figure 6, approach A1 has two building blocks, MMCRC (Mobile multi-camera remixing client) and WEM (Web-based event manager). As the name suggest, MMCRC runs on a mobile client and is used for uploading images and videos whereas WEM is the event manager that runs on a remote web server. The communication channel used between two blocks is HTTP.

In the hybrid technique, the task of content upload is done by these nodes serially. Approach A2 has three building blocks. Like approach A1 it has same MMCRC and WEM blocks but here they are not directly connected. They are connected via a middle ware block called PCMW (PC-based middle-ware). PCMW is a gateway, it passes all the information it received from MMCRC to the correct WEM. Communication channel used between MMCRC and PCMW is USB whereas HTTP is used between PCMW and WEM. In the hybrid technique, the task of content upload is performed by the MMCRC, PCMW and WEM connected serially.

3.1 Building Blocks

Building blocks used by approach A1 and A2 are described here.

3.1.1. Mobile Multi Camera Remixing Client

MMCRC provides a simple mobile interface to browse, preview and upload media of one or more event(s). It synchronizes videos/images and upload-status data (or meta-data) with PCMW. It also downloads or streams AVR video and user-contributed media files from the WEM. The upload mechanisms consist of approaches A1 and A2 described in section 3. In the A1 mode, the mobile device uploads directly to the WEM connected via any suitable transport (it could be cellular network or WLAN). MMCRC can be also used for event creation.

The design of MMCRC depends on the basic functionality of the node. Based on the functionality, the node consists of two main components a Media recoding client and MMCRC client. Media recording client is the native camera application on the mobile device that records media and saves image(s) and video(s) files to the predefined

location inside MC. For every recorded image and video files, a mobile camera application records a text file called context file. The context file stores meta-data of the recorded media file contains this information:

```
Unique identifier for MC
Image or video captured location
Recording time
Information needed by AVR creation system (which could be co-located with the WEM).
```

PCMW and WEM have the capability to manage multiple MMCRC nodes. Every MMCRC node needs a unique identifier. Along with media upload information inside context file also gets uploaded. The unique identifier stored inside the context file is used to recognize the media source MC on the WEM repository. Reference implementation uses the IMEI number of the MC as a unique identifier.

The context file also records media shoot location. Location information is used by MMCRC for organizing data whereas WEM can use it for various purposes such as organizing events, locating on world maps or resolving user search query based on location. The context file records time stamp of the shoot media. This can also be used for organizing media by MMCRC and WEM and also for resolving user search query based on date time. The context file can record media shooting information needed by the AVR algorithm. The MMCRC Client Application provides a user interaction interface for EBUTM on the MC. Basically the application provides interfaces towards PCMW and WEM for uploading selected media files. Tasks of MMCRC client application are grouped into four parts, Event Organizer, Event Browser, Event Search and Event Uploader.

The **Event Organizer** organizes camera captured images and videos based information recorded to context files. Organizer helps grouping media files (images and videos) present in the client device for segregation and selection of media to be uploaded. Event organizer maintains database for those events for which the user has contributed from the MC.

The **Event Browser** provides media browser to browse through the recorded images and videos. It also provides image viewer and video player (can use native players or

can provide its own) for media selection before upload. It downloads and list all events created by multiple users and stored at the WEM and provides existing events detailed information, which makes it easy for event selection. Detailed information is:

```
Event name
Event description
Event date
Shooting date
Shoot location
Number of contributors
Details about current user contribution
Number of media uploaded
```

MMCRC also provides detailed information of events created by current user. It also decodes and display search result received from WEM.

The **Event Search** engine was not in the scope of the thesis. The MMCRC application was using the search engine interface to request the list of events. It then receives and stores HTTP POST response to database. The received events are transferred to the Event Browser for response decoding and for displaying event information to user.

The **Event Uploader** provides WEM's interface for event creation and uploads media files for the new event or for the existing events to WEM in online mode. In case of a PC-based upload, event uploader creates a task list for PCMW in the form of job file. The job file stores detailed information of all the events and their media files selected for a PC-based upload. Later PCMW reads job files and takes a backup of media files mentioned in the job list along with context files. It then deletes the job files and informs MMCRC about completion. Event uploader then changes the status of the task accordingly. Job file for PCMW must list this information, MC unique identifier or IMEI number, login information if WEM is password protected, event list. Each event in event list has these properties:

```
Event id
Event name
Event description
Event Date
```

Path of the media file
 Media file type i.e. Image or video
 Other information

PCMW uploads event properties along with media files to WEM.

3.1.2. PC-based Middleware

A PC node is PC based middleware (PCMW) which acts as a gateway between remote web service and MMCRC for videos upload. PCMW reads media files and upload to WEM. It acts as a Gateway for EBUTM upload. This gateway is only used when the MMCRC client selects the PC upload option. Depending on the functionality, PCMW is divided into two sub components PC-MC connectivity and PCMW Host Application.

PC-MC connectivity is data communication between PCMW and MMCRC on top of Universal Serial Bus (USB) protocols. USB is an asymmetrical protocol. PC as a USB host initiates communication. In order to simplify the connection of USB devices, the USB communities have defined a number of standard protocols (called Classes) that the host software can support in a uniform manner. Microsoft has supported these classes since Windows XP. By using these Classes the target device will be able to link to the PC end without needing to provide any special drivers in the PC but using the standard Windows 7, XP and Vista software [\[13\]](#).

To make MC appear to the PC as a file system, a special driver is required. The drivers of MC are vendor specific and are easily available through the internet. For example, Nokia devices require Nokia Connectivity Cable Drivers which come with an OVI suite or PC suite installers. For example, to use a USB cable to connect the PC and the Nokia phone, appropriate Nokia Connectivity Cable Drivers required to be installed. For cable models DKU-2, DKE-2, CA-53, CA-70, CA-101, and CA-42 the drivers are installed automatically during Nokia PC Suite installation. Other Nokia cable driver setup programs can be found on the Nokia support pages [\[14\]](#).

These device specific drivers also expose Application Programming Interface (API) for PC based application to access the file system of MC. For example Nokia PC

Connectivity API is used to access device data from PC applications written in Microsoft Visual C++, Microsoft Visual C# .NET, or Microsoft Visual Basic .NET. Figure 7, shows the data flow between a PC and MC via a USB communication.

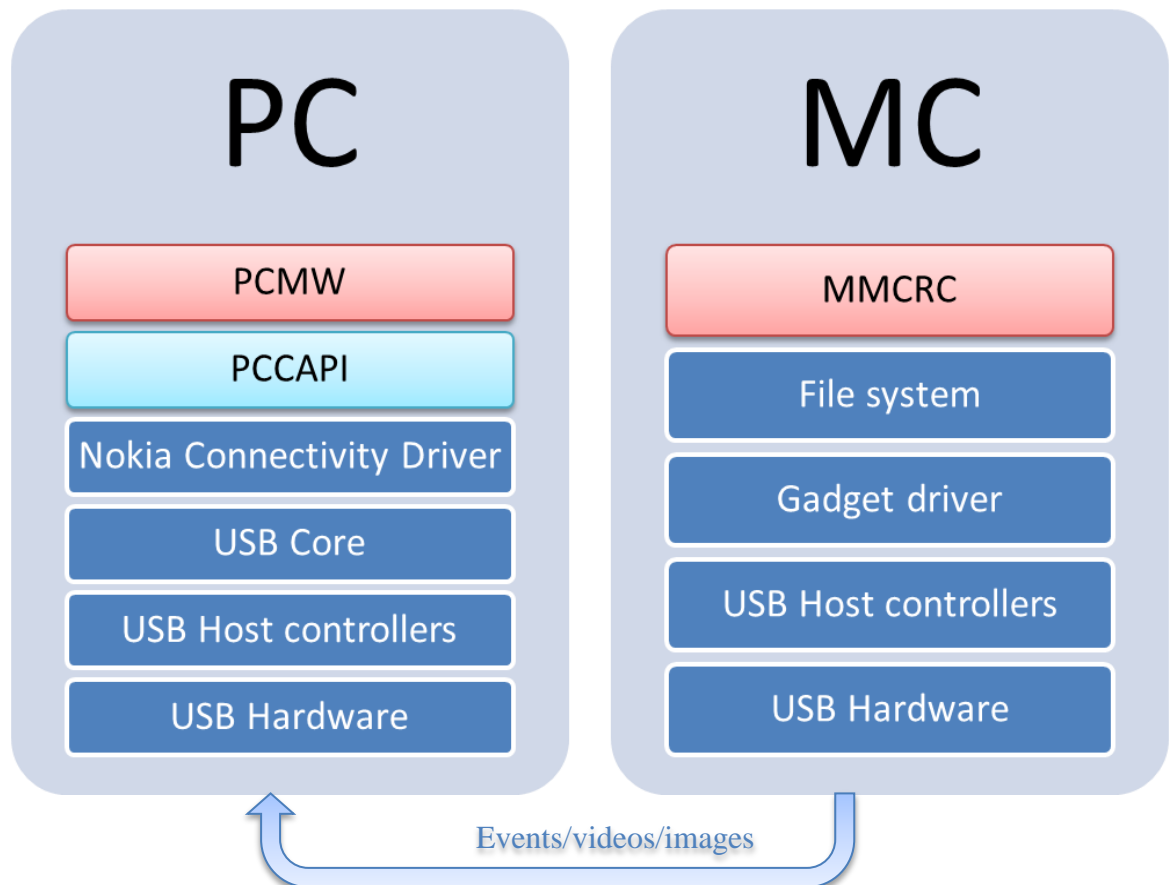


Figure 7: EBUTM Layers

As shown in Figure 7, there is no direct connection between PCMW and MMCRC components. MMCRC writes data to MC's file system and PCMW application accesses MC's file system by USB connection. PCMW and MMCRC are the components which are discussed in reference implementation chapter 4.

PCMW Host Application connects MMCRC and WEM. It transfer events created by MMCRC to WEM. Basic requirement of this host application to work is a USB connection between PC-MC and active Internet connection on PC. The tasks of PCMW application is grouped into two parts, Backup Manager and Upload Manager. Both managers work on the producer-consumer model.

Backup Manager does backup operation which starts after successful USB connection between PC and MC. Once the connection is made, Backup Manager moves the job file created by MMCRC to PC. Based on information in Job file Backup Manager creates separate directory for each event and moves media files along with the context file of the same event to the created directory. It then request Upload Manager for event upload and provides meta-data information read from Job file and location of the copied media. USB connection only needed till Backup Manager is active, once it finishes copying data, connection can be removed.

Upload Manager does not require USB connection, it just require information supplied by Backup manager for event upload. Upload Manager receives events information from Backup Manager and upload media files, context file and meta-data as HTTP POST request to WEM.

This study provides reference implementation of the given architecture. Reference Implementation provides detailed description of MMCRC and PCMW.

3.1.3. Web-based Event Manager

Web based event manager (WEM) is a functional entity that provides functions for media storage, AVR creation, video streaming, and networking service. An event can also be created directly using WEM. However the design of WEM is not within the scope of the thesis.

3.2 Communication Protocol

The information exchange between three nodes uses two communication protocols a HTTP and USB.

3.2.1. Hypertext Transfer Protocol

The **Hypertext Transfer Protocol (HTTP)** is a networking protocol for distributed, collaborative, hypermedia information systems. HTTP functions as a request-response protocol in the client-server computing model. In HTTP, a web browser, for example,

acts as a client, while an application running on a computer hosting a web site functions as a server. The client submits an HTTP request message to the server. The server, which stores content, or provides resources, such as HTML files, or performs other functions on behalf of the client, returns a response message to the client. A response contains completion status information about the request and may contain any content requested by the client in its message body [15].

In the context of the thesis, MMCRC and PCMW node does communication with WEM using the HTTP protocol. HTTP-based communication is used to upload and download event's media and metadata. The selected communication is bidirectional.

3.2.2. Universal Serial Bus

USB (Universal Serial Bus) is an industry standard developed in the middle of 1990s that defines the cables, connectors and protocols used for connection, communication and power supply between computers and electronic devices. USB is by nature an asymmetrical protocol, where only USB hosts initiate communication [16].

In the context of this thesis, USB based service communication is used between MMCRC and PCMW for media and metadata transfer for the selected event. Selected transfer is always one directional. In the Reference implementation MMCRC node is hosted by Symbian 3.0 based Nokia devices and PCMW node is hosted by Windows operating system based PC. To access content existing in the Nokia device, PCMW uses the services of the PC Suite Connectivity APIs. The PC Suite Connectivity APIs utilizes USB's high speed bulk data transfer service as a communication channel.

The **PC Suite Connectivity API 3.2** is a software interface released with Nokia PC Suite 6.84 and Starter Edition 6.84 (SE). It provides an easy way to develop applications including interoperability with Nokia mobile devices for third-party software vendors. The PC Suite Connectivity API 3.2 provides sophisticated interfaces for managing connections between Nokia mobile devices and PCs with the Microsoft® Windows operating system. It includes, for example, information about connected devices, Bluetooth pairing and services for accessing and managing files and folders in

the file system of the device. It also enables the installation of applications and includes services for PIM content management [\[17\]](#).

To be able to use the PC Suite Connectivity API 3.2 the following is need:

- Nokia PC Suite 6.84 or Starter Edition 6.84 (SE).
- A Nokia mobile device supported by Nokia PC Suite version 6.84 or the latest version.
- If Bluetooth connection is used, Bluetooth pairing must be settled by using the PC Suite Connectivity API or Nokia PC Suite. Bluetooth devices that are paired by using some other pairing software are not shown to the PC Suite Connectivity API. For more information please refer Bluetooth implementations supported by Nokia PC Suite from Nokia PC Suite User Guide.
- A development environment with Unicode support.

Function calls in the PC Suite Connectivity API 3.2 are modeled after Microsoft® Windows API counterparts to create a usage and behavioral experience that is familiar to the Microsoft® Windows application developer community. Because of special interoperability requirements some minor differences have been adopted.

The following points should be kept in mind while using PCCAPI:

- The majority of the function calls return an error code as the return value of the function. Exceptional cases are documented.
- The PC Suite Connectivity API 3.2 user is responsible for memory allocation and de-allocation unless otherwise documented. It is always recommended to use the API deletion functions for the structures if available.
- All character strings are Unicode strings. [\[17\]](#)

4 Reference Implementation

This chapter presents the reference implementation of the EBUTM system. The chapter begins by describing the architecture of the reference implementation of the MMCRC and PCMW. The description includes software entities and communication protocols used by MMCRC and PCMW. The reference design is based on requirements discussed in the chapter 3 and it will be shown how the requirements are met. This chapter also describes no of software and hardware used for reference implementation followed by detailed design of MMCRC and PCMW.

The software architecture of the reference implementation is very flexible and can be ported to many different environments.

4.1 Mobile Multi Camera Remixing Client

MMCRC is specific to Mobile client. It refers to the native camera application and MMCRC client application. A camera application captures videos and images and creates a context file for every captured media. It records various information like location of the shoot, information of the device, date-time etc. to the context file. This information will be used for organizing images and videos by MMCRC client application. The context file is also used by WEM for creating AVRs. MMCRC client application is the local interface for WEM, through this it is possible to browse event shared to WEM, It also provide easy mechanism for contributing new events and medias to WEM. In the online mode, MMCRC client application uploads media files of event to the WEM directly from device. In the offline mode MMCRC enables selection of media for PC transfer.

The reference implementation is available for MMCRC Client application. The context file generation is out of the scope of the thesis. Reference implementation will only use the generated context file for the captured videos or images.

4.1.1. Setup for Hardware and Software

This section describes hardware and software setup used for MMCRC reference implementation.

Hardware

Nokia N8 is one of the advanced camera phones in the market with 12 Mega-pixels Camera. It has superb details, natural colors, very low noise levels, excellent wide angle Carl Zeiss lens, powerful Xenon flash for a mobile phone, good quality HD video with stereo sound, fast autofocus, and digital zoom gives surprisingly good results in video. Nokia N8 runs on Symbian 3 operating system. MMCRC basically supports all Nokia phones running Symbian 3 operating system [\[18\]](#).

Testing is done on N8 because of its Camera. Nokia N8 has micro USB port, so Micro USB Data Cable will be required for quick connectivity between MC and PC.

The **Micro-USB Cable** is designed to charge smartphone rapidly and help preserve the life of battery. Cable can also be used to charge phone from PC and upload and download files between PC and phone. It fits easily and conveniently use for device [\[19\]](#).

Software

The idea of this study is not bound to any specific device or phone. It can be used on any mobile device having a camera. Implementation of MMCRC required widely used cross platform application and UI framework.

Qt was selected because it is a cross-platform application framework that is widely used for developing application software with a graphical user interface (GUI).

For setting up development environment, Qt SDK was used. It combines the Qt framework with tools designed to streamline the creation of applications for Symbian and Maemo, MeeGo (Nokia N9) in addition to desktop platforms, such as Microsoft Windows, Mac OS X, and Linux.

- **Qt framework** - Intuitive APIs for C++ and CSS/JavaScript-like programming with **Qt Quick** for rapid UI creation.
- **Qt Creator IDE** - Powerful cross-platform integrated development environment, including UI designer tools and on-device debugging
- **Tools and tool chains** - Simulator, local and remote compilers, internationalization support, device tool chains and more.

QML (Qt Meta Language or Qt Modeling Language) is a JavaScript-based, declarative language for designing user interface centric applications. It is part of Qt Quick, the UI creation kit developed by Nokia within the Qt framework. QML is mainly used for mobile applications where touch input, fluid animations (60 FPS) and user experience are crucial. QML documents describe an object tree of elements. QML elements shipped with Qt are a sophisticated set of building blocks, graphical (e.g. rectangle, image) and behavioral (e.g. state, transition, animation). These elements can be combined to build components ranging in complexity from simple buttons and sliders, to complete internet-enabled programs [21].

Design of MMCRC application design was separated into two parts, UI and Model. Model was done using Qt, C++ whereas UI part is done using QML.

4.1.2. Design

This section discuss about implementation design of MMCRC. The whole implementation is divided into two sections. Qt based C++ Model and QML based User interface.

Overview:

This section introduces components and behavior of MMCRC. Tasks of MMCRC are broken into following use cases:

1. Organize and displays images and videos into multiple groups based on context file information.
2. Download and displays existing events from the WEM
3. Searches and downloads WEM events by using search tags.

4. Creates event at WEM
5. Provides easy user interaction for images and videos browsing and playing
6. Event wise selection of images and videos and upload via WLAN or create store for PC upload.
7. Previews all user contributions.

Reference implementation of UI was done using QML. Task of QML layer is based on above use cases. These four sub components show the distribution of tasks: Event Organizer, Event Browser, Event Search and Event Uploader. The task distribution is shown in Figure 8.

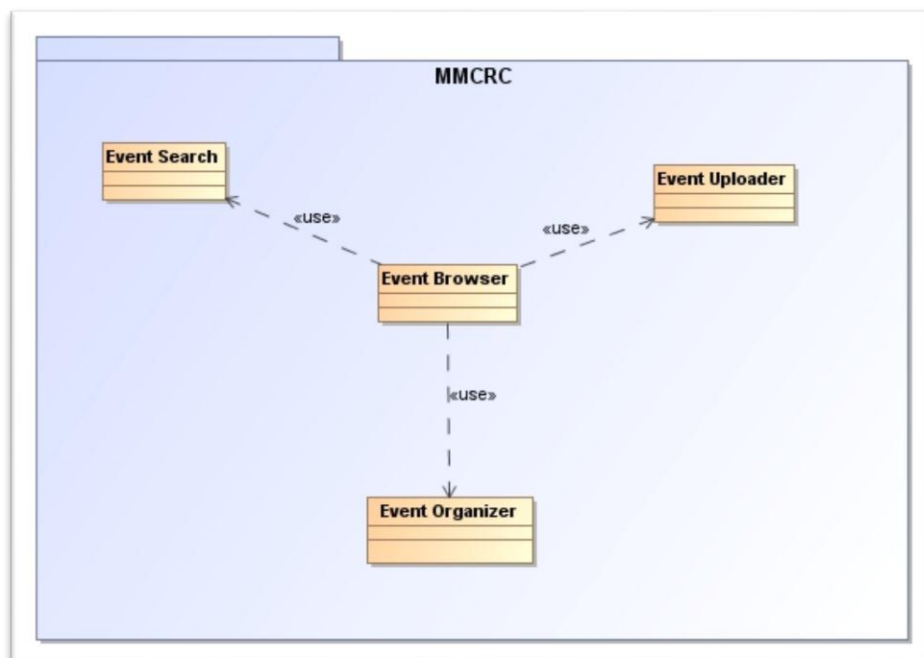


Figure 8: MMCRC Architectural Diagram

Camera application is used for capturing events in the form of images and videos. It record events and create context file for every captured media. **Event Organizer** searches for camera captured images and video files present in the devices. In Symbian devices, camera captured images goes under "*E:/Images/Camera*" and videos under "*E:/Videos/Camera*". Event Organizer, do recursive search inside "Camera" folder for ".jpeg", ".mp4" files and their context file. Qt C++ model stores these findings in structural ways, which is described in chapter 3. The Event Organizer does grouping of found images/videos based on creation date and event location. Media

recorded on the same day and at same location are grouped together. Recorded date and location can be found from context file. Event Organizer does not display all the event groups at one go to the user interface to avoid extra loading time. At maximum it fetches and displays a group of four events from latest to the oldest.

Once Event Organizer displays, group of four events, **Event Browser** takes the flow control. When the user selects one from four events, Event Browser with the help of Event Search, downloads matching events from WEM. The user can then browse downloaded events and their details. User can also do contribution to the selected event by uploading media from the selected organized group. It is also possible for the user to create new event at WEM by providing event details and later do media files upload from the selected organized group. Once an event gets created at WEM, it can be used by other users for media contribution and will be visible during their event search. The Event Browser provides an image and video viewer for the media files. Image and video viewer can be used for previewing before actually uploading media and AVR video(s). The Event Browser can be also used for browsing list of all events created and contributed by users in the past. The Event Organizer maintains the history of the events while the Event Browser provides facility for easy viewing the past contributions and to do another contribution.

The **Event search** does searching at WEM who maintains huge list of various kinds of events. Event searching logic was out of scope for the thesis. Only interface to the search was implemented as part of the thesis.

As the name suggest, **Event uploader** enables the creation of events in the WEM, uploads images and videos for the newly created or previously created events. The Event uploader also provides user a selection view for two kinds of uploading, active or passive. In the active uploading, it does uploading directly from MC client by finding available network connection present in the MC. The available network can be via WLAN or cellular network. In the passive uploading it creates job file for PCMW. It records all the events and their selected images and videos along with their event details.

The WEM access requires user authentication. The Event uploader takes user credentials and stores them in its database. In the case of an active upload, these credentials are also uploaded with uploaded data, whereas in the case of passive uploading, the user information is stored in the job file. An uploading a media file is a time-consuming task. The Event Uploader does it in background and keeps the UI informed of the upload percentage. The reference implementation of the MMCRC consists of several internal components which are listed:

File Manager: File Manager is the core of MMCRC. It manages C++ data model and perform requests from QML UI.

EventListModel: A model contains list of events information.

EventDetails: Event information's are represented by EventDetails. EventDetails list multiple file information

EventFilesModel: A model contains list of file information

FileInfo: File information's are represented by FileInfo.

XML Database Manager: It provides interface for XML database reading and writing.

XML Notifier: Notifies File Manager, when XML parser reads nodes from XML database.

Context File Reader: Contains information about camera captured media files.

Thumbnail Manager: Provide thumbnail image for the video files.

Uploader Thread: A separate line of execution for events and media files upload to WEM

DataUploadDevice: Provides mechanism for uploading video file of bigger sizes.

The **File Manager** is a main interface of C++ Qt model. It organizes images and video files present in the device. File Manager maintains a multi-hash table with event date as the key and FileInfo as the value. A FileInfo is user-defined data-type to store file path and information read from context file. Multi hash table is organized based on two criteria an event date-time and event location.

When an instance of File Manager is created during application launch, it searches for the camera captured image and video files inside "*E:/Images/Camera*" and "*E:/Videos/Camera*". It does recursive search inside these folders. On finding image and video files, it reads their context file and with the read values creates instance of the FileInfo data-type and stores it in the hash table. The Hash table is ready for use, once the search for images and video is finished. File Manager then organizes the hash table into a group of 4 events. Figure 9 shows the activity diagram, how to create group of four from the organizer list. The list of organized files can be huge and displaying whole list in on display, might affect the main thread execution. The UI view becomes very crowded, which makes it difficult for user to find the interesting one.

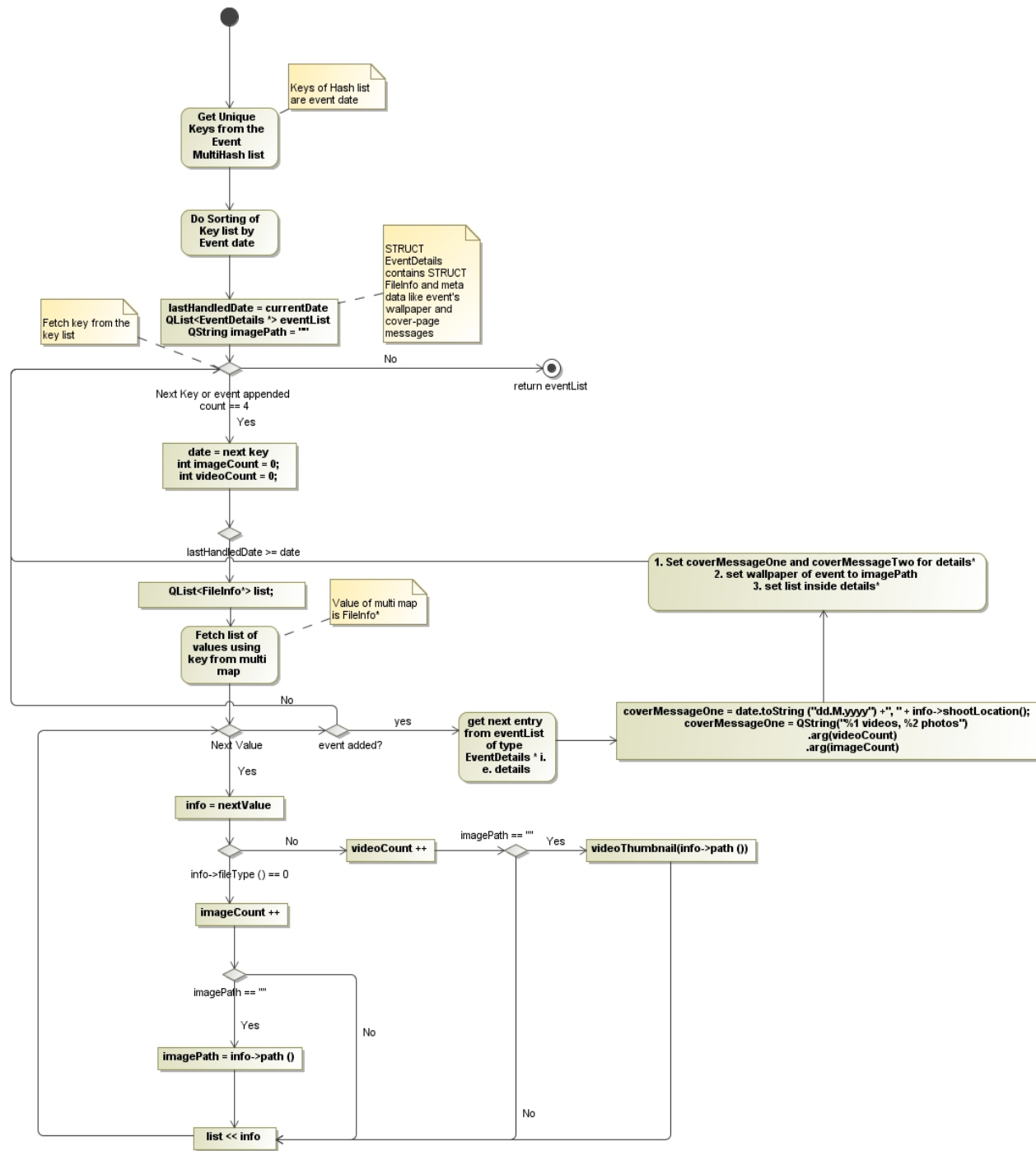


Figure 9: Flow Chart for Organizer

The File Manager also maintains a database listing all the contributions done by the user from MC. It also maintains database for the PCMW task list. On the application launch the File Manager reads these two databases and updates their data models. These models are later used by QML UI for display. The File Manager maintains three

data models. The first model represents events history where user has given his or her contributions, the second event list is for PC upload and the third model represents the event list for the WLAN upload. Figure 10 shows the basic structure of events models:

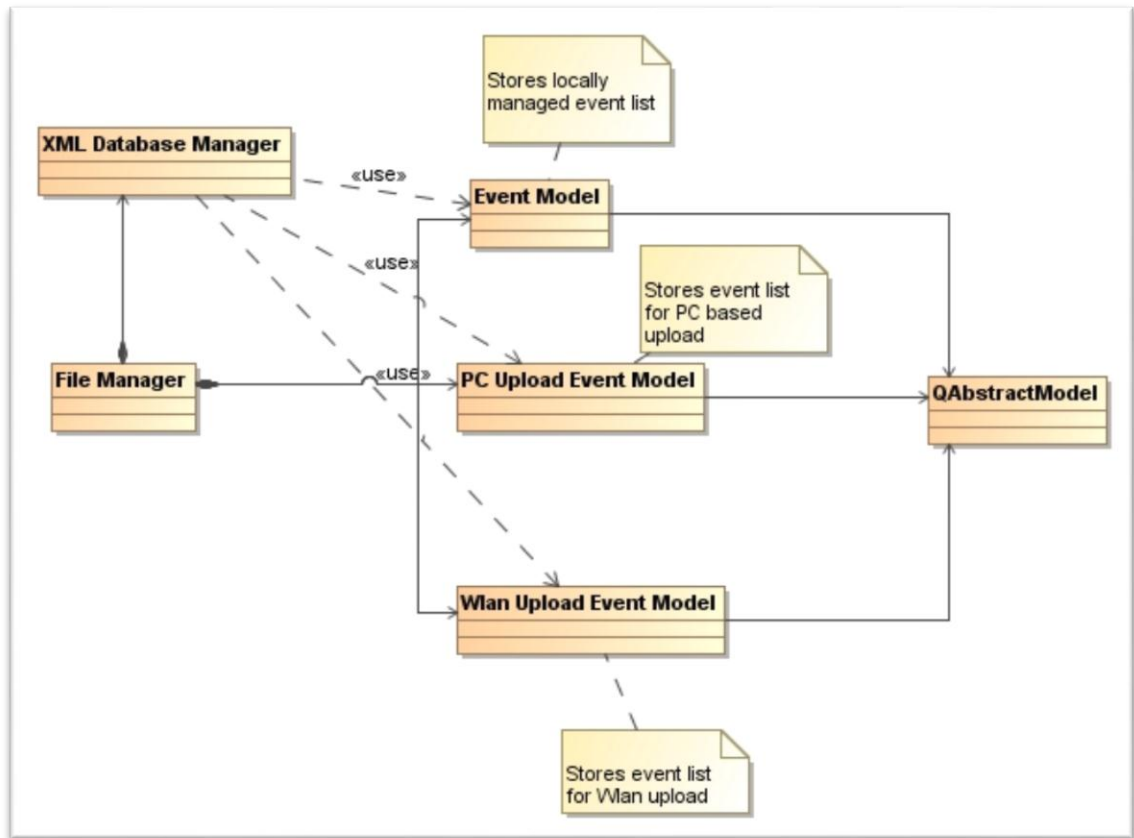


Figure 10: Models of XML Database Manager

The **EventListModel** is of type `QAbstractListModel`. Model contains a list of data type `QList`. It uses list to maintain events details. Information of each event is stored using data model of type `EventDetails`. The `EventDetails` is a user defined data type containing two types of properties: properties required by WEM and properties required by organizer of data type `MMCRCEventProperties`.

```

struct MMCRCEventProperties {
    /* date of event*/
    QString m_EventDate;
    /* name of event */
    QString m_EventName;
    /* description of event */
    QString m_EventDescription;
}
  
```

```

    /* wallpaper of event */
    QString m_EventWallpaper;
    /* Location where event was organized*/
    QString m_EventLocation;
    /* Upload message*/
    QString m_uploadMessage;
    /* WEM generated event id*/
    QString m_EventId;
    /*Selection mode used for event upload*/
    UploadSelectionType m_selectedType;
};

```

The organizer creates a group of 4 events. Each group is represented by user defined data-type **EventDetails**. Each group has dedicated wallpaper image and two messages. The first message contains organized date + shooting location where as the second message contains Image and video count for the organized group. The EventDetails also maintains a data model for image/video files. Model is of the user-defined type EventFilesModel. The **EventFilesModel** is of the QAbstractListModel type. The model contains a list of type QList and uses list to maintain media files of an event. Information of each file is stored using the type FileInfo. The **FileInfo** is a user-defined data type contains properties of image/video file. It contains property object called FileDetails.

```

struct FileDetails {
    /*Location of file, locally or remotely*/
    QString filePath;
    /* Upload mode selected for file*/
    UploadSelectionType selectedType;
    /*file type, video or image*/
    int type;
    /*Shoot position with respect to image of event stage*/
    QPoint shootPostion;
    /*last date of file modification*/
    QString lastModifiedDate;
};

```

The **XML Database Manager** is an interface to XML database. XML database file lists events and media files, authorization details, device details etc. XML Database Manager contains a XML parser for reading XML data files. File Manager listens to events generated by XML Database Manager after every read XML tag. File Manager updates its data model with the received data. These data models are used by QML based UI to provide history of event data. The history includes information of user contributed events, job files for PC transfer and pending jobs for WLAN transfer, waiting for network connection.

The user can update these models by adding new events or contribution to the existing events. QML UI sends these changes to File Manager, which gets forwarded to the XML database manager. XML Database manager writes those data back to XML data files.

Figure 11 describes the execution flow of the interaction between File Manager and XML Database Manager. XML Database reading is a onetime process, executed during application launch. By reading XML Database Manager makes the C++ model ready with the history and status of events.

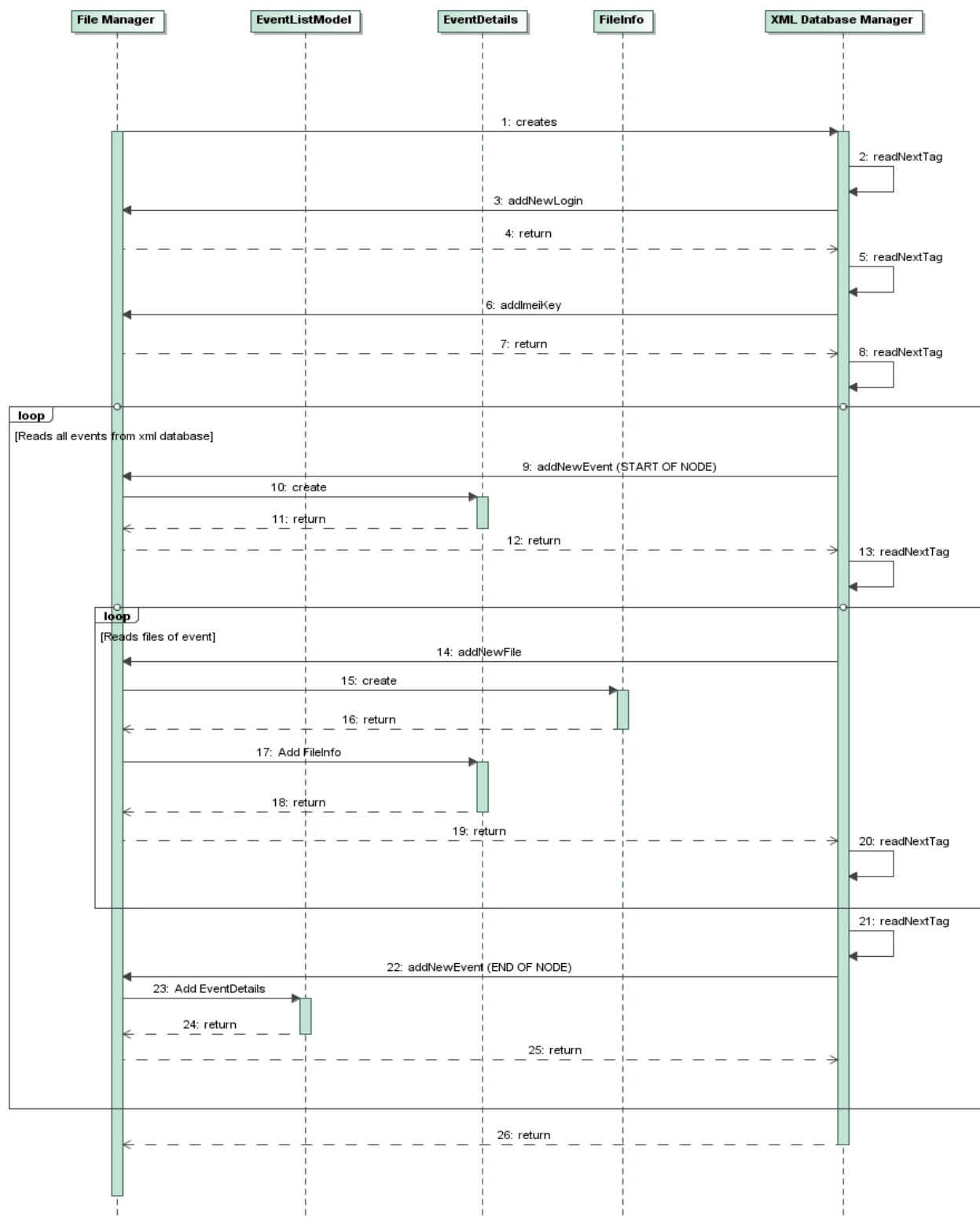


Figure 11: Model Management by XML Database Manager

XML database writing can occur at any time during the lifespan of an application. Whenever the user does contribution to events, the result status goes to the XML database. There are three situations where event status written to XML database:

event creation, file uploads via MC and file uploads via PC. There are two XML databases maintained: one contains all activities during above three situations called master copy and another XML database maintains only a task list selected for the PC upload. Figure 12 displays the procedure used for PC Task list creation.

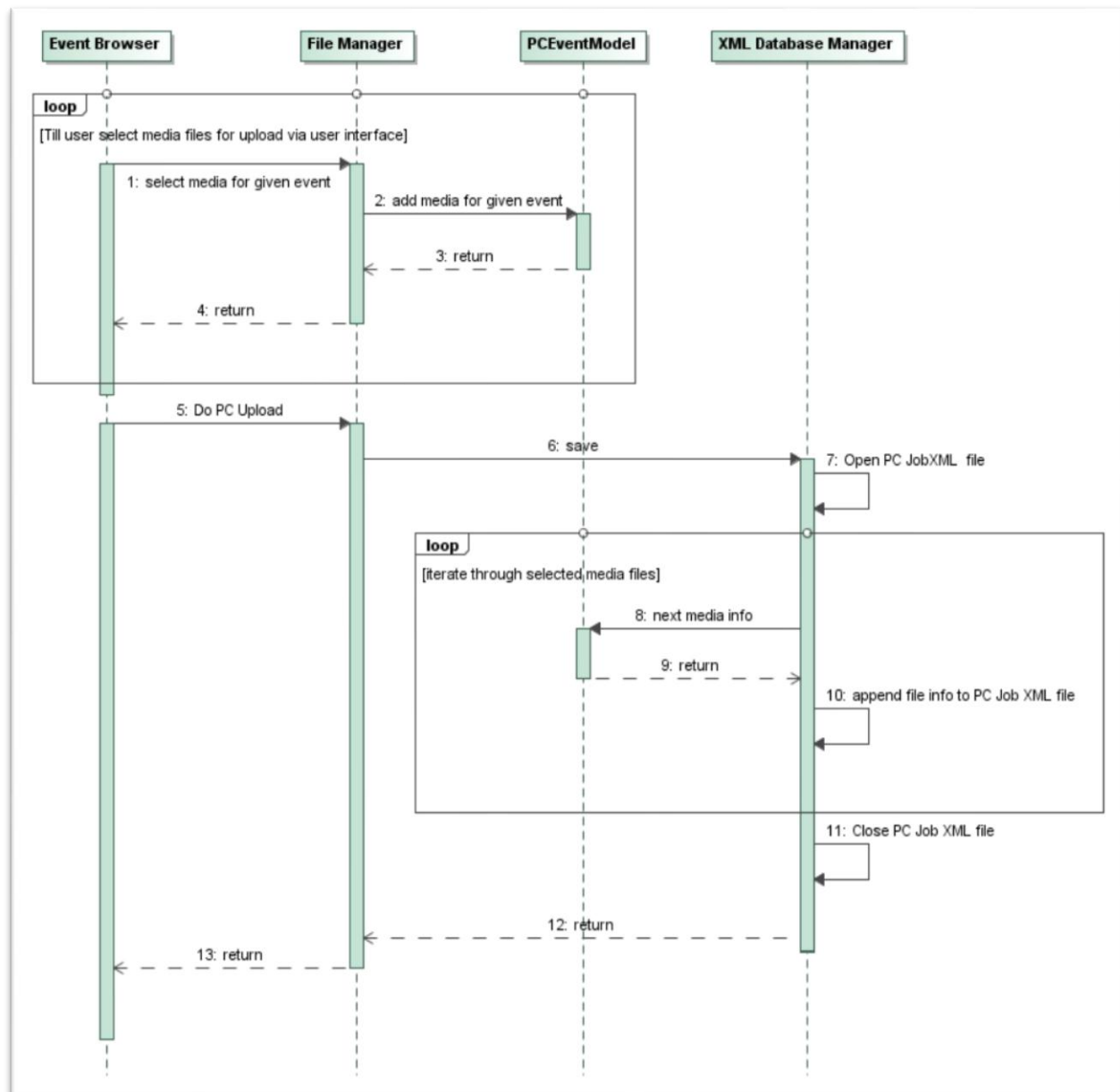


Figure 12: Job Creation for PCMW

During parsing of the XML data files, the **XML Notifier** is used by XML Database Manager to notify File Manager. Notification rose when XML parser finds these tags inside XML file such as

- 1) Login: login tag contains, Username and Password used for WEM authorization.
- 2) IMEI: IMEI tag contains IMEI number of the MC.
- 3) Event: event tag contains event details and it as multiple sub tags named file. Unlike login and IMEI, these tags can be of multiple counts.
- 4) File: Event tag can contains multiple file tags. Each file tag represents video/image and their details.

For these tags, XML Notifier provides notification methods. These methods are implemented by File Manager to get notified. Sample XML data file look like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<settings>
<login username="priyan" password="priyan"/>
<imei key="imei352684040156254.txt"/>

<event id="25" name="Pink Floyd concert" description="Friends
events" date="2011-08-01 00:00:00"
wallpaper="http://wem.com/events/25/thumbnail?api_key=priyan"
uploadMessage="Waiting for PC connection">

    <file path="E:/Images/Camera/201104/201104A0/11042011185.jpg"
type="0" selectionType="3" shoot_positionX="20"
shoot_positionY="20"/>

    <file path="E:/Images/Camera/201104/201104A0/11042011186.jpg"
type="1" selectionType="1" shoot_positionX="40"
shoot_positionY="40"/>

    <file path="E:/Images/Camera/201104/201104A0/11042011187.jpg"
type="0" selectionType="3" shoot_positionX="20"
shoot_positionY="40"/>
</event>
```

```

<event id="25" name="Madonna Concert" description="Attended in
Helsinki" date="2011-09-01 00:00:00"
wallpaper="http://wem.com/events/27/thumbnail?api_key=priyan"
uploadMessage="Uploaded via WLAN">

  <file path="E:/Images/Camera/201104/201104A0/11042011188.jpg"
type="0" selectionType="3" shoot_positionX="20"
shoot_positionY="20"/>
</event>
</settings>

```

For every camera captured Images and video files there is a context file storing information of the event. Context file can contains lots of other information based on various different use cases of application or WEM. This is a text file, read by **Context file reader**.

Context file naming convention is: <filename>+<device imei>.txt.

For example "11042011183.jpgimei352684040156254.txt"

On finding context file, Context file reader parses the name and finds the IMEI of device. Reading IMEI is a onetime process, once context manager finds IMEI from one context file then it ignores for the rest of the context files. It then opens context file, parses the details and provide it to File Manager.

QML UI needs to display thumbnail image of found video and image files. QML UI scale down image file to get required thumbnail size. It can't do the same for video files. To get thumbnail image of the video files, it request Thumbnail manager via File Manager. **Thumbnail manager** creates a thumbnail image of the videos and store it physically on device.

Thumbnail creation requires some image processing; it is wise not to repeat this process on every application launch or video browsing. Thumbnail manager creates thumbnail of requested video for the first request and later request just provides the stored link to the File Manager and saves extra processing. In reference implementation, this is the only platform specific component of MMCRC. There is no

support for thumbnails in Qt/Qt Mobility. MMCRC uses Symbian specific “CThumbnailManager” APIs to access it.

CThumbnailManager is the main interface class to thumbnail engine. Thumbnail engine can be used to request thumbnails for different kinds of media objects. The client using thumbnail engine must implement the MThumbnailManagerObserver. The observer interface is used to provide thumbnail data when it is available. The process of thumbnail creation is described in Figure 13.

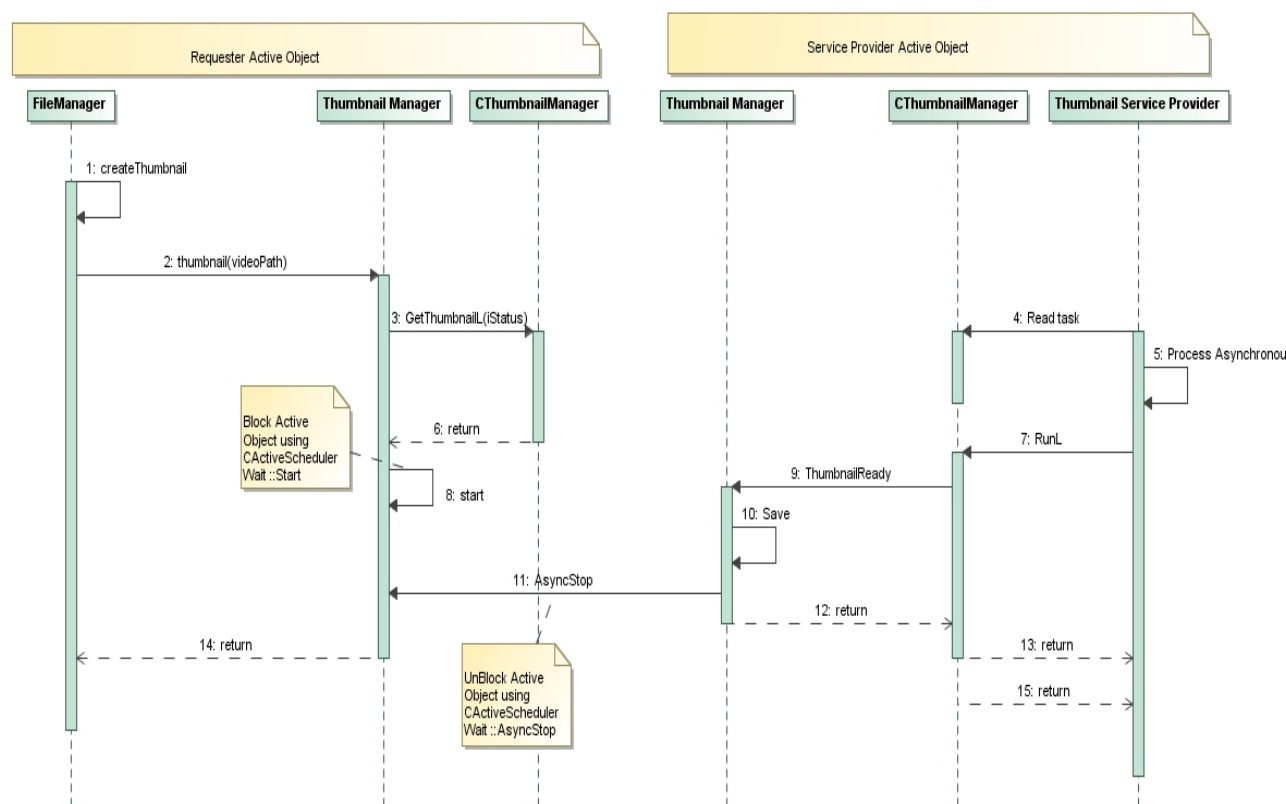


Figure 13: Thumbnail Manager

The uploading data to WEM can be a time consuming operation, Main thread under which QML UI is running cannot be used for this. File Manager creates an **Uploader Thread** for uploading task. The Uploader Thread represents a separate thread of control within the program; it shares data with main thread within the process but executes independently in the way that a separate program does on a multitasking operating system. The Uploader Thread interacts with WEM on behalf of File Manager. It does two tasks: event creation and media uploading.

Description of Figure 14:

```

1) Start
2) Create a new event
3) Upload event
4) Check Uploader Thread, if not started then start.
5) Send 'uploadRequested' signal to Uploader Thread.
6) Main thread goes to event loop.
7) 'doEventUpload' is a slot for 'uploadRequested' event in
   Uploader thread's event loop.
8) Check for network connection
9) Create and send Http post message to Http thread
10) Uploader thread goes to event loop for next upload request
    from main thread.
11) Http thread sends request to the remote web server and
    listen for the response.
12) Once all data sent to remote server, received 'finished'
    signal.
13) Http thread sends 'finished' signal to Uploader Thread.
14) Event loop of Uploader Thread receives 'finished' signal.
15) Http thread's event loop goes for next request and
    'onUploadFinished' slot catches 'finished' signal.
16) Uploader Thread informs main thread about upload completion.
17) Main thread's event loop receive upload completion event.
18) Uploader Thread's event loop goes for next request.
19) 'handleEventUpload' slot handles upload completion event.
20) End

```

Uploading media file is a time consuming process and require separate thread for upload. Uploaded file count can grow at any level so as processing. Process of uploading media does good usage of heap memory. The heap memory is first use for reading file and creating buffer out of it. Buffer also includes media meta-data information. The heap memory is also needed for doing multi-part HTTP::POST.

For devices like N8, camera captured images of size ~1MB or small sized videos can be easily managed on the heap memory, but bigger sized video upload can be little

difficult with the same approach because recorded video sizes can grow up to gigabytes. If the file size grows beyond 64 MB boundary for devices like N8, data upload heap buffer cannot be created including file content and their meta-information in one go. Thread will end up in low memory condition. The Uploader thread considers limitation of heap size of Symbian device. It takes different approach for images and video uploads. For images it creates an upload buffer and parses file and meta-data information together and uploads with Multi part HTTP Post. For videos, an Uploader thread uses services of **DataUploadDevice** to read file in 16K iterations and do multi part HTTP upload. File upload post message buffer contains:

```
Content of image/video file
Content of context file
User name
Password
Event id for which image is getting uploaded
Event name
File description
File type (image or video)
X-coordinate of the shoot location with respect to stage image
Y-coordinate of the shoot location with respect to stage image
```

Figure 15 shows the interaction between threads during media file upload. It also shows how processing of the Uploader thread changes with the change in media type.

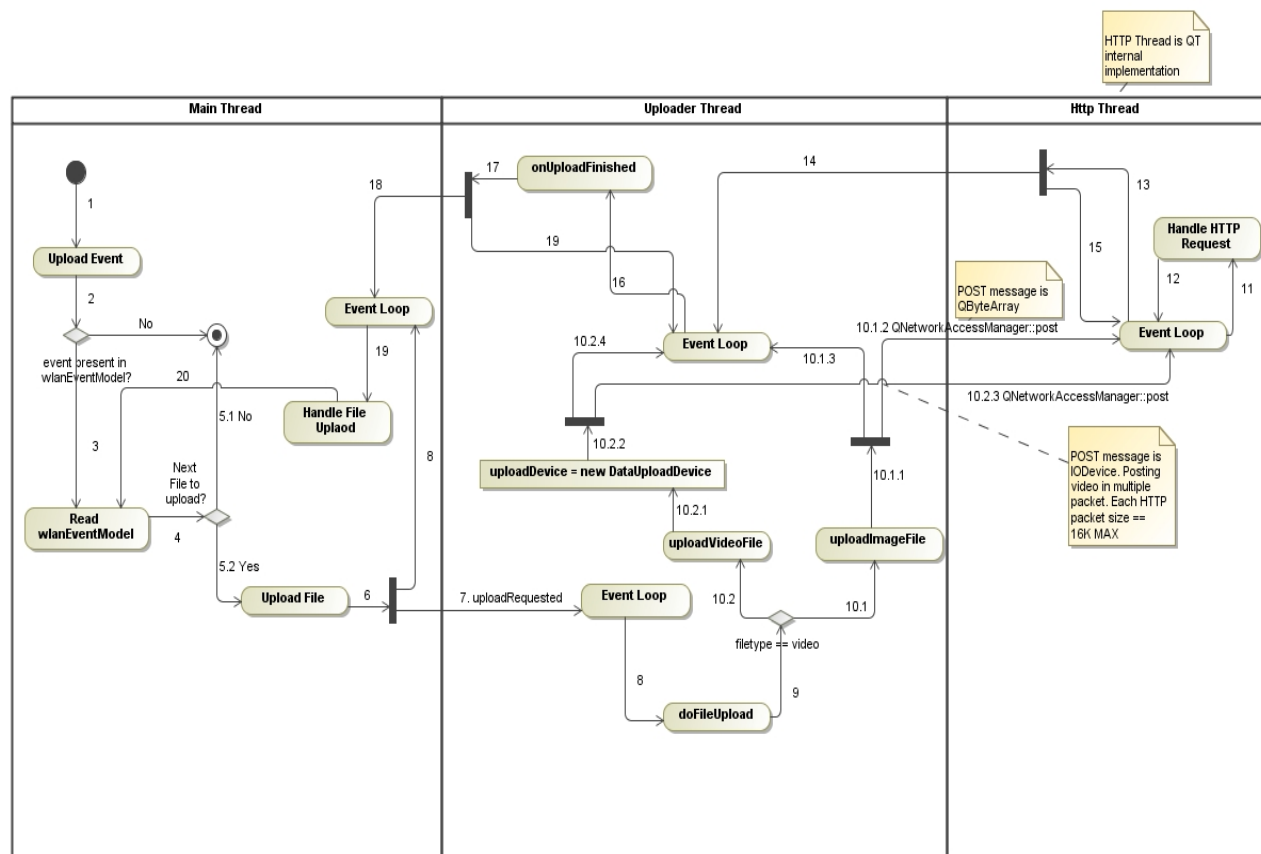


Figure 15: MMCRC's Thread interaction model for file uploading

During upload of the media files, the Uploader thread keeps on posting event to the Main thread about the uploaded percentage. The File Manager receives upload percentage and updates QML UI.

DataUploadDevice is of type QIODevice. The QIODevice class is the base interface class of all I/O devices in Qt. The QIODevice provides both a common implementation and an abstract interface for devices that support reading and writing of blocks of data, such as QFile, QBuffer and QTcpSocket. QIODevice is abstract and cannot be instantiated, but it is common to use the interface it defines to provide device-independent I/O features. The DataUploadDevice class operates on a QIODevice pointer, allowing them to be use file data sequentially with 16K iteration.

During video upload, Uploader thread does not create HTTP::Post message buffer by itself, it just creates an instance of a DataUploaderDevice and provide its pointer to QNetworkAccessManager by calling this method:


```
QNetworkReply *post(const QNetworkRequest &request, QIODevice
*data);
```

The HTTP framework while doing multipart post uses QIODevice data pointer for message reading. QIODevice calls readData() method implemented by DataUploadDevice. The Uploader thread divides task for DataUploadDevice into three parts: a message header, file content and message footer. The QIODevice gives a callback readData() after availability of 16K free buffer. The DataUploadDevice fill this buffer first with message header then read files in parts and then message footer. Once the DataUploadDevice finishes the message footer, it closes the media file and gets ready for another task from the Uploader thread. The DataUploadDevice implements four methods of abstract class QIODevice:

```
/*Before accessing the device, open must be called for media
file*/
bool openFile();

/*This method closes the media file once copied successfully to
buffer*/
void closeFile();

/* Reads up to maxSize bytes from the device into data, and
returns the number of bytes read. Here max size is 16K
*/
qint64 read ( char * data, qint64 maxSize );

/*
Does nothing just a place holder
*/
qint64 writeData(const char *data, qint64 len);

/*This function is called by QIODevice, when memory available
for file reading. Internally this method called above mentioned
read method for file reading.
*/
```

```
    qint64 readData(char *data, qint64 maxlen);

    /*calculate size of the file + upload headers or media meta-
    data*/
    qint64 size() const;
```

4.2 PC-based Middleware

PCMW is PC based middle ware component which contains MC-PC USB connectivity interface and PC based client application for backing up images/videos from MC to PC and then upload to WEM.

4.2.1. Setup for Hardware and Software

This chapter describes hardware and software setup used for MMCRC reference implementation.

Hardware:

The idea of PCMW components are not bound to any platform. It can run on any platform which does not have battery, processing and memory constraints. Microsoft® Windows-based personal computer or laptop is selected to reference implementation due to the highest number of users reach. Micro USB data cable is used to connect Microsoft® Windows based Personal computer and Nokia N8. To access Nokia N8 file system, Nokia PC connectivity driver is required to be installed. It comes with Nokia OVI-suite, which can be found and install from http://www.comms.ovi.com/m/p/ovi/suite/index_en_uk.html. All installation instruction can be found from web page. After installation, file system of the Nokia N8 can be accessed from Microsoft® Windows based PC.

Software:

PCMW client application required development environment which can be easily portable to other platforms. Qt a cross-platform application framework is chosen as it is supported for desktop platforms, such as Microsoft Windows, Mac OS X, and Linux. Qt is a comprehensive application and UI framework for developing Windows applications that can also be deployed across many other desktop and embedded operating systems without rewriting the source code. To give file system access to Nokia N8, Nokia PC Connectivity API is used. The Nokia PC Connectivity API (PCCAPI) enables developers of PC applications to add device-connectivity capabilities to their applications for Series 40 and S60 devices supported by Nokia PC Suite. The Nokia PC

Connectivity API (PCCAPI) is part of Nokia PC Suite and takes advantage of the suite's existing capabilities. It's been designed to free application developers from the need to deal directly with the complexities of the connectivity and transmission protocols used by Nokia PC Suite and the need to understand the architecture of connected devices. Nokia PC Connectivity API can be installed from:

http://www.developer.nokia.com/Resources/Tools_and_downloads/Other/PC_Connectivity_API/

PCCAPI is MFC library, built using MSVSC compiler comes with Microsoft Visual Studio 2005, 2008 and 2010. So Microsoft Visual Studio 2005, 2008 and 2010 also needs to be installed. By default Qt uses, MinGW as compiler, but reference implementation needed MSVC2008 to compile Qt based PCMW application. Reference implementation uses, Microsoft® Windows based PC as a hardware platform, so PCMW client application is Windows application. Qt SDK for Microsoft Windows required to be installed as a development environment. Installation Setup and guide can be found from <http://qt.nokia.com/products/platform/qt-for-windows/> . Qt supports a wide range of Windows platforms.

The following software tools are required on a Windows development system.

- Microsoft* Visual Studio 2008* express edition or above
- **Qt SDK version 1.1.2** for Windows
- Nokia OVI suite
- PCCAPI

Qt Creator IDE is used as development tool, which comes with Qt SDK. The Qt SDK bundles the MinGW compiler, then get a version of Qt built with MinGW. By default Qt uses the MinGW compiler and program will be built using it. While installing Qt libraries 4.7.1 for Windows (VS 2008, 228 MB), configure Qt Creator to use that set of Qt tools/libraries to build applications. The "qmake" from that set of libraries should use VS2008 by default. To do this from Qt Creator, follow these instructions:

- Select Tools menu, Options, and then Qt4.
- Locate the "qmake" from the bin directory of VS2008 version of Qt
- Give the new Qt instance a meaningful name

4.2.2. Design

This chapter discusses about implementation design of PCMW. The whole implementation is divided into two sections: a Qt based C++ Model interfacing PCCAPI and QML based UI.

Overview

This chapter introduces components and behavior of PCMW. It implements following use cases: establish USB connection with MC, move MMCRC written XML based task list from MC's store to PC's store, copy image and video files from MC to PC store and upload image and video files from PC to WEM store. The reference implementation of the PCMW consists of several internal components which are listed in this chapter. Detailed descriptions about each component are described as a sub chapter in this document. The internal components of PCMW are:

File Manager: File Manager is the core of PCMW. It manages C++ data model and perform requests from QML UI.

PCMWEventProperties: A structure contains properties of events.

Device Handler: It's a handler manages connection with MC and do backup of the media files to PC from MC with the help of XML Database reader.

XML Database Reader: XML Database reader reads XML task list file and inform Device Handler using XML notifier.

XML Notifier: Notifies File Manager, when XML Database Reader reads nodes from XML task-list file.

IDMAPIDeviceNotify: A PCCAPI interface, which informs File Manager about status of MC PC Suite connection.

DeviceReaderThread: A thread provides separate line of execution for copying files from MC.

FilesHandler: A handler created in the context of DeviceReaderThread for copy data over the connection created by DeviceHandler.

FileTransferProperties: A structure which contains properties of files to be copied from MC. It is passed between main thread to DeviceReaderThread.

FilesUploader: A class which is used for uploading files copied from the MC. It works as producer consumer mechanism with DeviceHandler. Where producer is device Handler and consumer is FileUploader.

FilesUploaderThread: A thread provides separate line of execution for uploading files from PC to WEM.

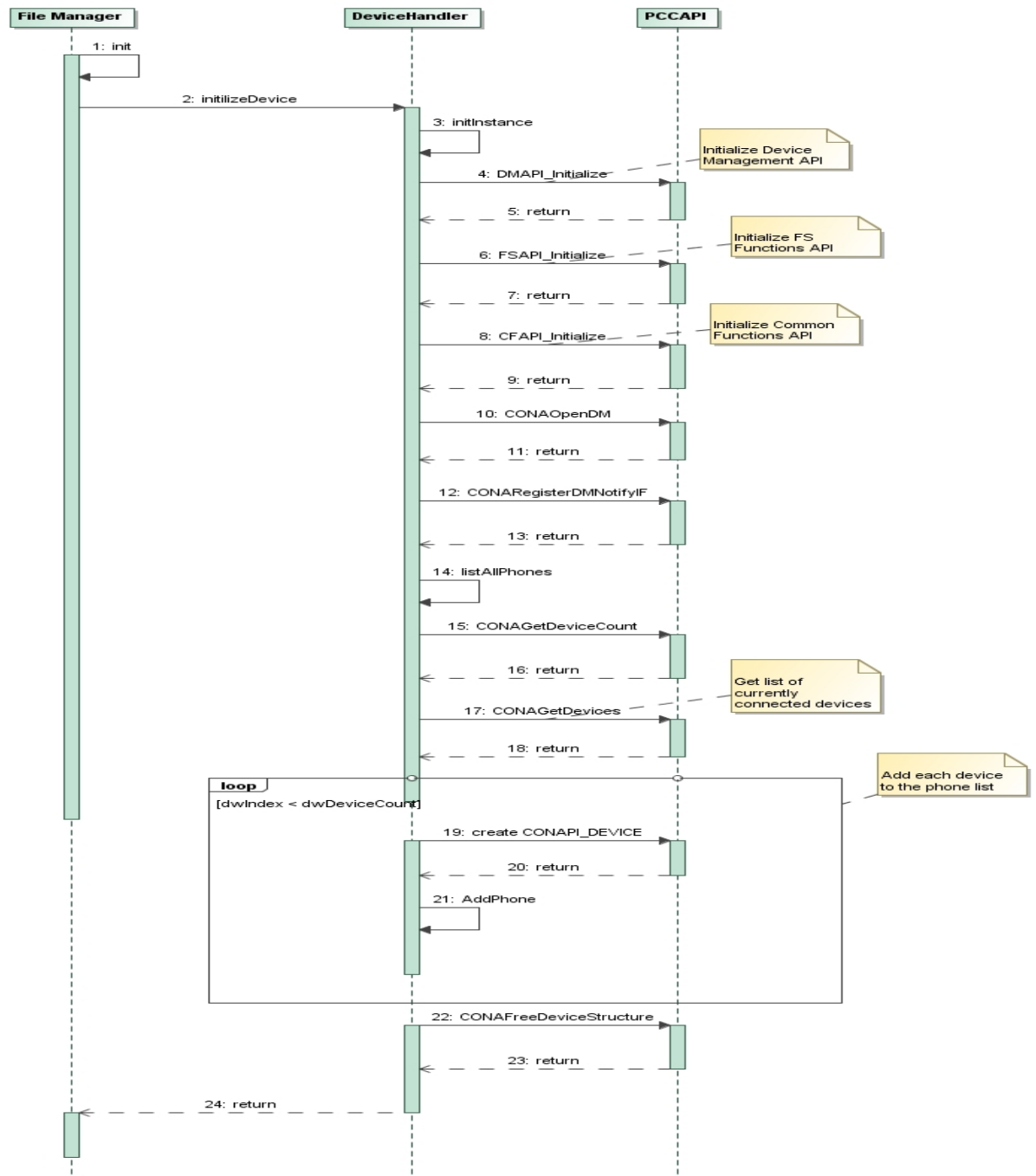
HttpFileUploader: A handler created in the context of FilesUploaderThread for uploading data over established Http connection with WEM.

FileUploadInfo: A structure which contains properties of a file to be uploaded by HttpFileUploader. It is passed between DeviceReaderThread and FilesUploaderThread.

DataUploadDevice: Provides mechanism for uploading video file of bigger sizes.

File Manager is main interface of C++ Qt model, all requests from QML based UI comes via File Manager. The File manager uses services of other elements of class diagram for completing job created by the QML UI layer. The File Manager creates PC

suite connection using PCCAPI interface with the USB connected MC. It maintains list of all MC devices connected in PC suite mode. The MC devices are uniquely identified using their serial number. Figure 16 shows steps used to initialize connection between PCMW client and MC's file system.



Using PCCAPI interface File manager accesses MC's file system. The current reference implementation uses first enumerated MC device found during initialization of PC Suite connection. After successful device connection, the File Manager moves XML based task file to PC. As discussed in chapter 4.1, XML based file contains detail list of events needs to be uploaded by PC. The File Manager with the help of Device Manager copies files selected for WEM upload, organize these files and update their data list model with event details. These event details are later used by the File Uploader during file uploading. Once all files are copied, the Device Manager closes the device connection and informs File Uploader to start uploading. An operation of copying files from MC and uploading files to WEM are synchronous as well as asynchronous. File Manager provides both kinds of API support. These APIs are also used during performance analysis of uploading.

In a synchronous approach, the device manager runs under main thread first completes data copying from MC then informs the Uploader thread which later completes files uploading to WEM. The Uploader thread also informs main thread about progress of file upload. In an asynchronous approach, the device manager and File uploader runs in separate thread. Here the Device Manager and File uploader follows producer-consumer model. The main thread requests the Device Manager to start making connection with MC. The Device Manager thread notifies main thread once it is done with connection establishment and finished with Job file copying from MC to PC. After notification, Device Manager copy event files from MC file system. In parallel main thread invokes Uploader thread for file uploading. Device Manager and Uploader thread shares a common task list buffer. The Device Manager generates a task by copying media and its context file from MC file system and putting it into the shared buffer then notifies waiting Uploader thread and then repeat the same steps for another media file. Then Uploader thread wakes up with the notification and starts uploading files (media and context file) and event meta-data information to WEM. An uploading is done with multi-part http post as done by MMCRC. The Uploader thread uses the MMCRC mechanism for video file with the help of DataUploadDevice where it uploads video file and event meta-data into 16K sizes. Both Device Manager and Uploader thread informs main thread about uploading status. The main thread display progress of MC file copy and WEM file upload via user interface.

The **Device Handler** interfaces PCCAPI for managing PC Suite connections with multiple MCs. It opens the channel with the MC's file system and provides access to files in it. The basic job of the Device Handler is to take back-up of media files from MC's to a PC. The backup operation starts with copying XML based task list file. With the help of XML Database reader, the Device Handler creates a map list with file path in MC as key and value as FileUploadInfo structure containing detailed information of the file including location where file to be copied in PC. After finishing XML file, information the map list is used for copying files from MC to PC. Key of the map list is taken as file location in MC whereas destination location is decided based on the value inside instance of FileUploadInfo. The Device Handler provides synchronous and asynchronous API. In synchronous API it does their task in the context of main thread as described in the chapter 4.1. In an asynchronous approach, thread called DeviceReaderThread does task in cooperation with the Uploader thread to achieve maximum efficiency.

The **XML Database Manager** contains a XML parser for reading XML data files. With the help of interface called XML Notifier, the Device Handler listens to events generated by the XML Database Manager after every read xml tag. The Device Handler updates its map list with the received data. The map list is later used by Device Handler for files copy from MC.

The Device Handler with the help of **DeviceHandlerThread**, does file copy in parallel with data upload. The DeviceHandlerThread creates instance of File Handler in the context of new thread. The File Handler contains pointer to Device Handler, copies files from MC independent of the main thread. On every file transfer the Device Handler sends "fileTransferCompleted" event to File Manager and updates the result status. File manager later update status to User Interface. At the end of file copy, Device Handler also sends "mobileToPCTransferCompleted" event to tell all file transfer from MC to PC are completed. File Handler provide generic interface for PC-MC connectivity, Device Handler implements this interface and give PCCAPI specific connectivity solution.

The File Manager instantiates a **FileUploaderThread** to create an independent line of execution for HTTP file uploading. The FileUploaderThread creates instance of FileUploader in the context of new thread. The FileUploader contains instance to FileManager, using it uploads file from PC to WEM. The File Uploader sends

"uploadProgress" event to FileManager during upload then sends "filesUploaded" event after completion of every file upload. The File Uploader also sends "pcToWEMTransferCompleted" event when all file transfer to WEM gets completed. The FileUploader provides generic interface for file uploading, the HttpFileUploader provides HTTP specific solution for file uploading and classify uploading technique according to file type. For bigger size files (video files), it splits file contents into multi parts. At maximum each part can contains 16K of size. The HttpFileUploader and DataUploadDevice behave exactly same way as MMCRC's UploaderThread and DataUploadDevice mentioned in chapter 4.1.

5 Upload Performance Measurement

The upload performance of the EBUTM is the crucial indicator for the overall system performance. The reference implementation has been designed to test and analyze performance of EBUTM from different angles. Result of the performance test provides data which is useful in judging suitable uploading technique for various sizes of media file(s).

This chapter presents the results about the time required for uploading media files. The focus has been on end-to-end performance. The measurements include upload from mobile device and PC both following both techniques of EBUTM [A1](#) and [A2](#).

The basic purpose behind this measurement was to analyze which of the EBUTM technique is better suited for uploading media from mobile devices. In order to find the measurement results, both MMCRC and PCMW applications were used.

Data upload testing for **MMCRC** was done using WLAN connection between Mobile phone and ADSL Modem and 3.5G network. Whereas data upload testing for **PCMW** was done over WLAN connection between PC and ADSL modem and Laptop connected with ADSL modem via Ethernet cable. However upload impact due to CPU utilization not used during EBUTM performance analysis.

5.1 Measurement Setup

These are the configuration of used hardware for testing:

ADSL Modem: Zyxel P-660HW-D1 is used for connecting to the internet. ADSL (Asymmetric Digital Subscriber Line) is a high speed internet access service uses standard telephone lines to transmit upstream and downstream data on a digital frequency. The "asymmetric" in ADSL refers to the fact that the downstream data rate, or the data coming to computer from the Internet, is traveling faster than upstream data, or the data traveling from your computer to the Internet. Upstream data rates are slower because web page requests are fairly miniscule data strings that do not

require much bandwidth to handle efficiently. Model used for ADSL connection supports the following data rates: the downstream data rate: 1015-9952 kbps and the upstream data rate: 16 to 640 Kbps [24].

Laptop: HP 8530w running PCMW connected to ADSL Modem using Ethernet cable and WLAN [23].

- Intel® Core(TM) 2 Duo Core T960 @2.80GHz
- RAM: 4 GB
- Operating system: 64bit Windows 7 Enterprise

MMCRC runs on Nokia N8 connected with 3.5G network and with ADSL modem via WLAN [18].

- 3G HSDPA, 10.2 Mbps, HSUPA, 2.0 Mbps
- WLAN Wi-Fi 802.11 b7g7n, UPnP technology

5.2 Test Files

Video recording is done using Nokia N8 for different durations starting from video clips of short duration and going towards long duration video clips (the video resolution was 720p for all the cases). Same files are used for uploading directly from mobile (approach A1) and subsequently via Laptop (approach A2). Table 1 shows the recording time and size of the used test files:

Table 1: Record duration v/s File size

Recording Duration(in minutes, MM:SS)	File Size(MBs)
0:02	3
0:10	10
2:24	206
3:01	235
5:01	425
7:19	578
10:02	812
13:34	976
15:06	1178

20:05	1600
-------	------

As stated in the table just 20 minutes of high definition video clip requires 1.6 GBs of memory. It will go up in the case of full HD video (1080p) recording.

5.3 Test Results

Test files are uploaded using networks mentioned in section 5.1. Measurement was done for uploading time (in minutes). Output data was plotted on graph to show the uploading trend. Received output was plotted into two graphs:

Graph A: Uploading time (Y axis) v/s File size (X axis)

Graph B: Uploading time (Y axis) v/s recording time (X axis)

Graph A plotted between uploading time(Y axis) v/s file size (X axis). This is to analyze which EBUTUM technique is better when the file size grows.

Table 2: Upload time v/s upload time

File size (MBs)	3	10	206	235	425	578	812	976	1178	1600
Upload-time(MMCRC-WLAN) in seconds	1.01	1.93	41	41.68	79.33	101.65	155.76	180.18	207.21	287.21
Upload-time (PCMW-PC) in seconds	0.55	1.86	35.25	39.28	72.45	96.73	139.73	158.95	189.61	274.66
Upload-time(MMCRC-3.5G) in seconds	6.28	10.96	158.96	184.58	224	298.63	450.15	522.93	683.66	828.31

Figure 17 shows a liner graph by substituting file size from Table 2 for x-axis and upload time for y-axis.

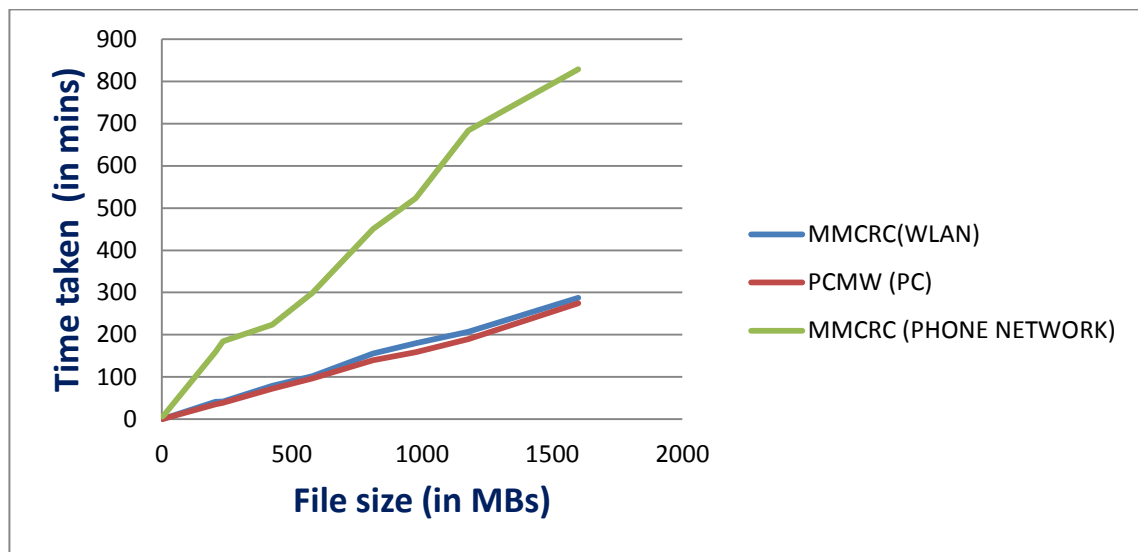


Figure 17: Uploadtime v/s Filesize

Figure 17 shows how uploading trend changes for same file size with different upload approach. It is also visible that uploading time difference by using phone network compare to WLAN/PC grows with the increase in file size.

Graph B plotted between uploading Time (Y axis) v/s recording time (X axis). This is to analyze, which EBUTUM technique is better as recording time increases.

Table 3: Uploading time v/s recording time

Recording time	0.033	0.16	206	235	425	578	812	976	1178	1600
Upload-time(MMCRC-WLAN)	1.01	1.93	41	41.68	79.33	101.65	155.76	180.18	207.21	287.21
Upload-time (PCMW-PC)	0.55	1.86	35.25	39.28	72.45	96.73	139.73	158.95	189.61	274.66
Upload-time(MMCRC-3.5G)	6.28	10.96	158.96	184.58	224	298.63	450.15	522.93	683.66	828.31

Figure 18 shows a liner graph by substituting recording time from Table 3 for x-axis and upload time for y-axis.

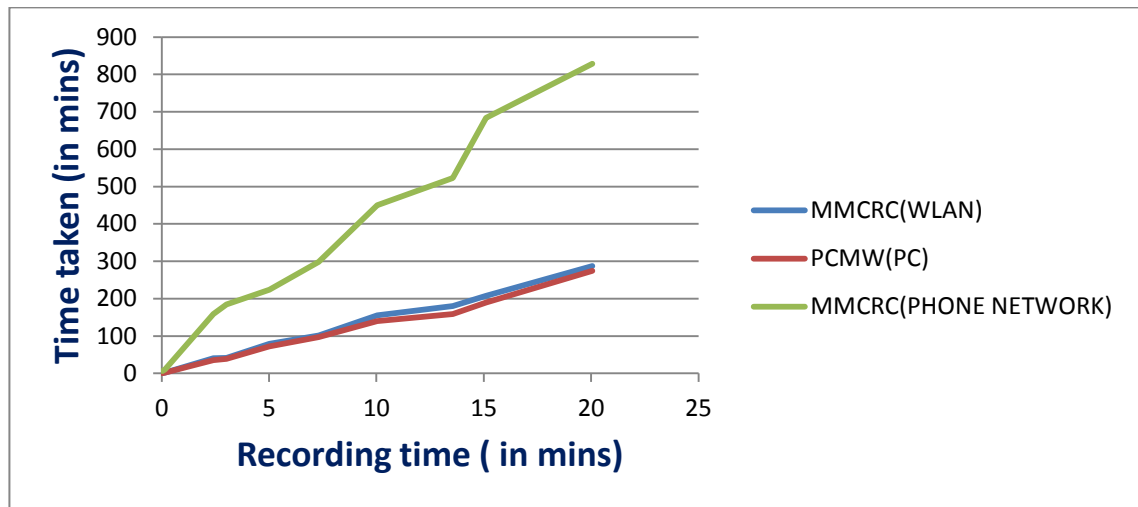


Figure 18: Uploadtime vs Recordingtime

Figure 18 shows how uploading changes trend for same recording time with different upload approach. It is also visible that uploading time difference by using phone network compare to WLAN/PC grows with the increase in recording time.

5.4 Result Analysis

This section of research is here to determine what is found during upload analysis. Provided graph in chapter 5.3 will allow here to give conclusion about each uploading techniques. It allows analyzing which uploading technique would be better considering upload time and factors like battery life and uploading cost.

Upload analysis was done with one of the most frequently used modem i.e. ADSL modem. ADSL model has limitation in upstream data rates. It follows better the internet connection speed, the faster downloading capability. Configuration of used ADSL model is: Internet Service Provider: Elisa oyj and average downlink speed is: 0.86 Mbps. The local link upload speed between device and Modem is: 0.99 Mbps [25].

On analyzing graphs shown in chapter 5.3, it is evident that in terms of uploading time there is not much difference between PC and Mobile (WLAN) whereas upload via phone network is not practical with the upload algorithm tested during study. In order to assess uploading performance and to judge best uploading technique required benchmarking criteria.

Here on the basis of uploading time and battery life of mobile device the user has to decide suitable uploading method. On drawing a limit where the user can upload up-to 1 hour using mobile device, then it is not practical to use mobile like PC by connecting it via charger always and locking device for a sole purpose and forcing it to a stationary device. As recording time grows bigger, locking period needed to complete the upload grows bigger too. It's better to put these videos in the queue of PC upload instead of uploading directly from mobile device. Another limit we will draw i.e. cellular network will be used maximum 10 minutes for uploading. Beyond this limit, cost of uploading can be prohibitive (depending on the subscription plan) and upload via WLAN or PC can be used.

With these two criteria, these results can be drawn: Upload via phone network can be a suitable option for file size maximum of 10 MB or video file with recording time of 10 seconds. Upload via mobile phone with WLAN connected can be a suitable option for file size maximum of 680 MBs or video file with recording time of 8.5 minutes. Upload via PC is suitable for anything above 680 MB files or videos recording time of above 8.5 minutes (at 720p video resolution). The drawn result can further enhanced by considering these factors:

1. The experience of capturing videos and pictures at the event is fresh right after the event and becomes less clear with time. Thus it is more beneficial for the user and the service to make use of the content as early as possible.
2. Variability in the amount of content captured.
3. Availability of network connectivity and bandwidth
4. Battery power consumption rate of device
5. Cost of data uploads
6. User movement
7. CPU usage for upload affecting other application performance

6 User Feedback

A restricted group of people tried EBUTUM technique by using MMCRC and PCMW application, based on their feedback, some major issues are identified and conclusions can be made.

A positive user experience is a critical part of the overall user experience and helps defining success of the product. Usability is the foundation of a successful user interface, and is the easiest process for creating a positive user experience. User experience of MMCRC application was key to the success of the study. A set of questions was asked to 10 users, their answer revealed some interesting and encouraging results.

Details about test participants:

In 10 users, 4 were between 20-30 years of age whereas 6 were of 30-40 age range. Between them 2 were female and rest of them were male. Whereas 2 were beginner and rest was advance in terms of technical level.

Asked questions are of two types, first 7 questions for selecting a choice from the given lists whereas 4 questions about rating.

These questions were asked to select best suitable option:

1. Do you like to contribute media files to web service(s), yes or no?

The result of the question is plotted and shown as Figure 19, which says 60 percentage of user likes to contribute media file to web services.

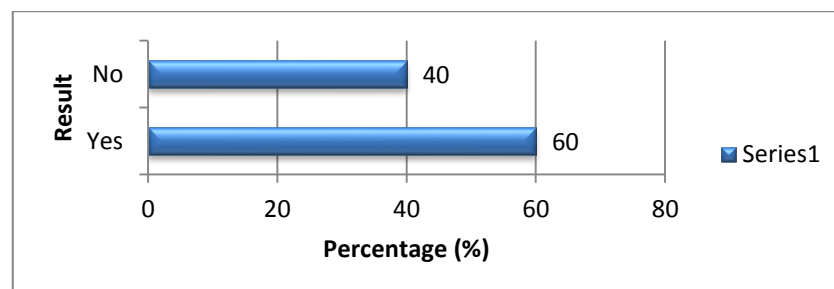


Figure 19: Question 1

2. Do you like shared contribution compared to individual?

The result of the question is plotted and shown as Figure 20, which says 70 percentage of user likes to do shared contribution.

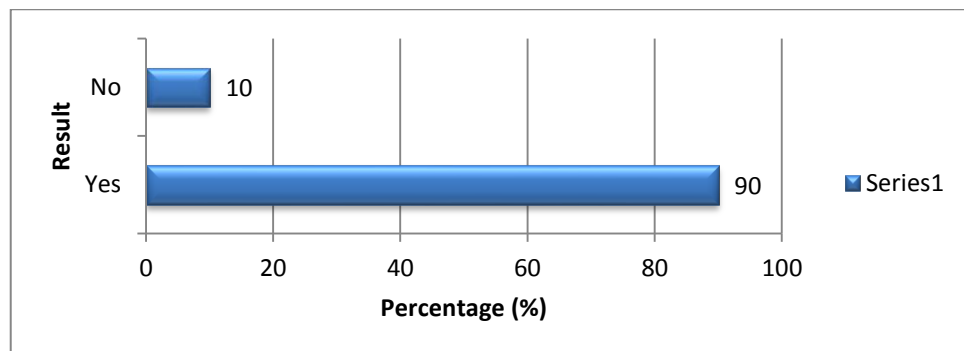


Figure 20: Question 2

3. Do you mostly use mobile for recording and sharing?

The result of the question is plotted and shown as Figure 21, which says 60 percentage of user do recording and sharing using mobile phone.

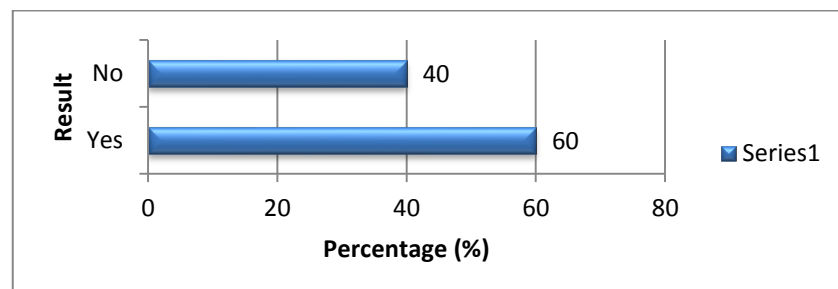


Figure 21: Question 3

4. Do you like to get AVR video(s), yes or no?

The result of the question is plotted and shown as Figure 22, which says 90 percentage of user would like to get AVR videos.

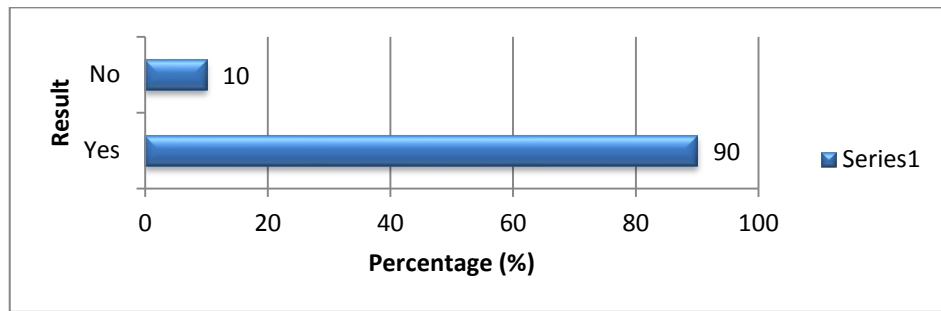


Figure 22: Question 4

5. Do you like to contribute media files just after the event or later?

The result of the question is plotted and shown as Figure 23, which says 60 percentage of user like to update media files just after the event recording.

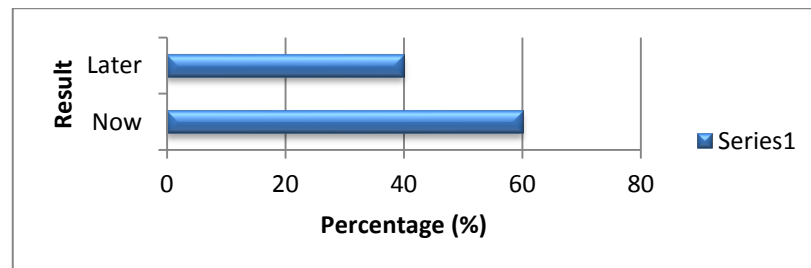


Figure 23: Question 5

6. Which technique you would like to use, mobile (MMCRC) or pc (PCMW) upload?

The result of the question is plotted and shown as Figure 24, which says 65 percentage of user like to use PCMW approach.

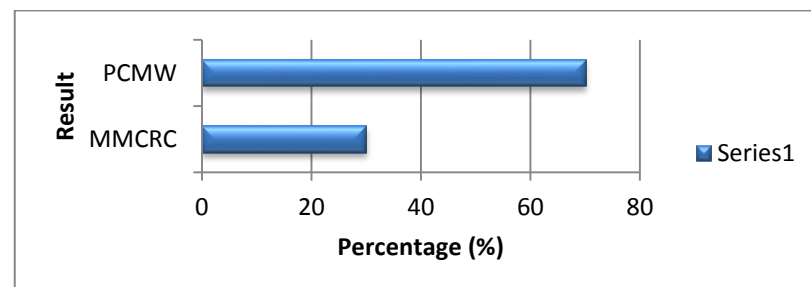


Figure 24: Question 6

7. Do you like to know suggestions/hints by the client application, which technique is more useful based on type of selected media file, yes or no?

The result of the question is plotted and shown as Figure 25, which says 60 percentage of user like to get hint for best suited uploading technique.

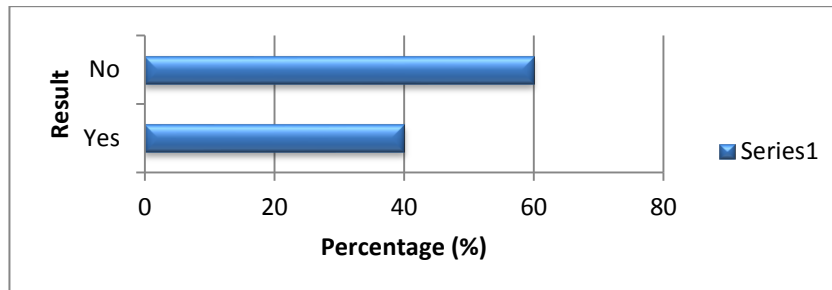


Figure 25: Question 7

These questions were asked to rate the following between value 1-5:

	1	2	3	4	5	
poor	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Excellent

1. Do Event based uploading technique looks useful? Please rate.

The result of the question is plotted and shown as Figure 26, which says mostly people feel good about event based uploading technique.

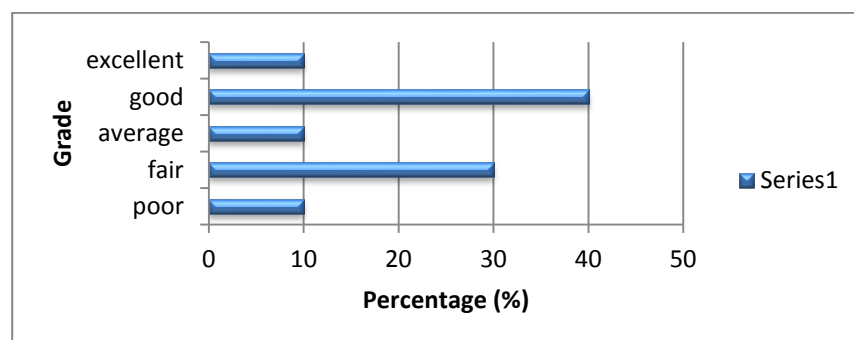


Figure 26: Question 8

2. Do event based uploading technique saves time and resources? Please rate.

The result of the question is plotted and shown as Figure 27, which says mostly people feel event based uploading technique saves time and resources.

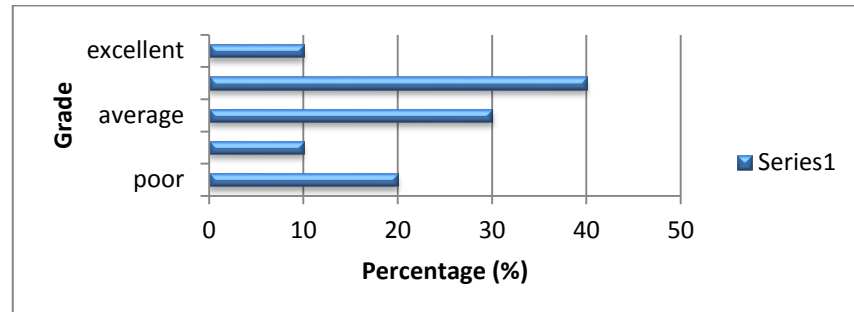


Figure 27: Question 9

3. Did you like user interface of MMCRC application? Please rate.

The result of the question is plotted and shown as Figure 28, which says mostly liked user interface of MMCRC client.

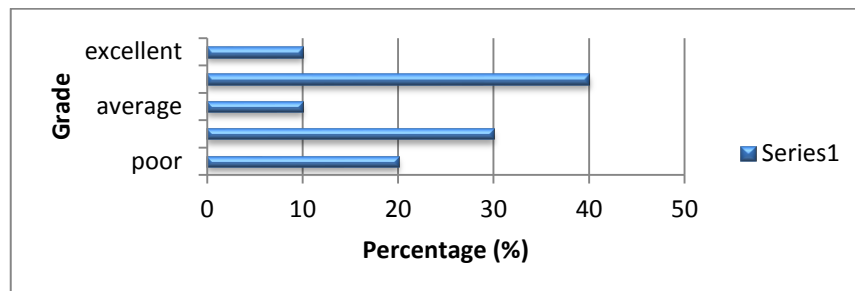


Figure 28: Question 10

4. Is this applied media selection technique (before uploading) is interactive and easily understandable?

The result of the question is plotted and shown as Figure 29, which says mostly liked event selection techniques used in MMCRC application.

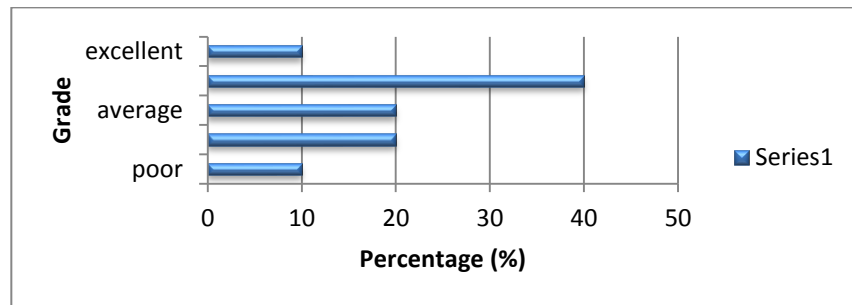


Figure 29: Question 11

Every person in the group agreed that the principle idea is good one, and they gave their resolute approval, though they need some improvement and addition in the principal idea.

At the end of the usage, smile on the face of user creates a reason to believe that there is hope for the idea to be used in the future.

7 Conclusions

A goal of this project was to understand the nature and requirements of event participants in order to provide suggestions for the best possible event sharing mechanism. Also to analyze conditions best suited to use mobile devices for uploading and conditions suitable to use PC for upload. The goal was achieved by introducing EBUTM uploading technique, which was tested by developing MMCRC and PCMW client applications. Upload analysis has been done with real time video recording and uploading to web services. Our study tried finding the best methods for uploading images and videos recorded by mobile phones. The conclusion made was there is no single method which is better in all use cases of uploading. In the process of analysis it was found that it was not just the fast uploading that matters but uploading cost and battery consumption rates were also benchmarking factors. The thesis emphasizes the fact that mobile devices are basically meant for voice call and used to get the maximum mobility, to do videos and images uploading mobile device can't be made a stationary device.

The thesis presented a solution for content-based event sharing and proposed a user friendly approach to upload content from mobile devices. It encourages end-user participation in social media services, by enabling content upload at minimal latency. Social events are generally recorded by many attendants. For each attendant it is difficult to cover every important moments of the event. Hence uploading incomplete picture of the event may not be very wise idea. Our study gave emphasis on first uploading recorded images and videos to central location and then sharing a maximum coverage of the event in the social network. It also highlighted the idea of sharing less content but good content. Our study suggested running video editing/merging logic at central location server to combine all the uploaded video and images into one master copy called remixed video. The chances of master copy (AVR) to be better and better informative are more compared to scattered copies. This makes master copy a better candidate for sharing in the social network.

In the end user feedback matters most. The user feedback was done after demonstrating end to end EBTUM technique with the help of MMCRC, PCMW application and WEM running as web services. Generic feeling of the user feedback was encouraging. Test participants also wanted the whole system to be platform independent and also more intelligent. They wanted EBUTM to provide uploading hints.

In the future, the reference implementation can removes platform specific solution used for displaying video thumbnails from the MMCRC application. After that the MMCRC application can run on any QT based platform just with minimal form factor adjustments.

References

- 1 Taniar David, Ma Jianhua etc, Journal of Mobile Multimedia [online];
March, 2005
URL: <http://www.rintonpress.com/journals/jmm/jmm-leaflet.pdf>
Accessed: 6 October, 2011

- 2 What is Event? [online], Event
URL: <http://www.en.wikipedia.org/wiki/Event>
Accessed: 8 October, 2011

- 3 Jayson Sharon, A few wrinkles are etching Facebook, other social sites [online];
15 January, 2009
URL:
http://www.usatoday.com/printedition/life/20090115/socialnetworking15_st.art.htm
Accessed: 6 October, 2011

- 4 Voskresensky Mitya, Content is the fuel of social web [online]; 6 May 2011
URL: <http://www.slideshare.net/duckofdoom/aol-nielsen-content-sharing-study>
Accessed: 12 October, 2011

- 5 Speaking of Events [online], Event Management
URL: <http://www.juliasilvers.com/embok.htm>
Accessed: 8 October, 2011

- 6 Kaltura [online], Open Source Video Solutions
URL: <http://www.kaltura.org>
Accessed: 8 October, 2011

- 7 Vihavainen Sami, Mate Sujeet, Seppälä Lassi, Cricri Francesco,
Igor D.D. Curcio; We Want More: Human-Computer Collaboration in Mobile
Social Video Remixing of Music Concerts; May 2011
URL:
http://www.hiit.fi/~svihavai/chi2011_vihavainen_wewantmore.pdf
Accessed: 12 January, 2012

- 8 Igor D.D. Curcio, Vinod Kumar Malamal Vadakital, Miska M. Hannuksela;
Geo-predictive real-time media delivery in mobile environment; Oct 2010
URL: <http://dl.acm.org/citation.cfm?id=1878036>
Accessed: 12 January, 2012
- 9 Igor D.D. Curcio, Singh, Varun, Vinod, Kumar Malamal Vadakital;
Method and apparatus for providing a Geo-predictive streaming service;
September 2010
URL: <http://www.freepatentsonline.com/y2012/0009890.html>
Accessed: 12 January, 2012
- 10 Igor D.D. Curcio, Kontola, Kalervo Mikael, Hannuksela, Miska Matias,
Vinod, Kumar Malamal Vadakital; Predictive bit-rate modification of content
delivery in a wireless network; October 2009
URL: <http://www.freepatentsonline.com/EP2347629.html>
Accessed: 12 January, 2012
- 11 File Allocation Table [online]
URL: http://en.wikipedia.org/wiki/File_Allocation_Table
Accessed: 11 October, 2011
- 12 Social media impact on revenue – 8 statistics from financial services,
InvestmentPal [online]; 20 July 2011
URL:
<http://blog.investmentpal.com/social-media-impact-on-financial-services-revenue/>
Accessed: 11 October, 2011
- 13 What is USB? [online], Universal Serial Bus
URL: <http://en.wikipedia.org/wiki/USB>
Accessed: 16 October, 2011
- 14 Nokia PC Suite [online], How to use Nokia PC Suite
URL:
<http://europe.nokia.com/support/product-support/nokia-pc-suite>
http://nds1.nokia.com/files/support/global/phones/software/Nokia_PC_Suite_in_stallation_eng.pdf
Accessed: 15 October 2011

- 15 What is HTTP? [online], Hypertext Transfer Protocol
URL: http://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol
Accessed: 11 October, 2011
- 16 Embedded USB - a brief tutorial [online]
URL:
http://www.computersolutions.co.uk/info/Embedded_tutorials/usb_tutorial.htm
Accessed: 11 October, 2011
- 17 PCCAPI [online], Nokia PC Suite Connectivity API – Features
URL:
http://www.developer.nokia.com/Resources/Tools_and_downloads/Other/PC_Connectivity_API/Features.xhtml
Accessed: 2 November, 2011
- 18 Nokia N8, Gsmarena [online]
URL: http://www.gsmarena.com/nokia_n8-3252.php
Accessed: 13 November, 2011
- 19 Nokia Charging and Data Cable CA-185CD, Micro USB Cable [online]
URL:
<http://europe.nokia.com/find-products/accessories/sales-package-accessories/nokia-charging-and-data-cable-ca-185cd>
Accessed: 12 November, 2011
- 20 What is Qt? [online], Qt
URL: <http://qt.nokia.com/products/>
Accessed: 10 November, 2011
- 21 What is QML? [online], QML
URL: <http://en.wikipedia.org/wiki/QML>
Accessed: 10 November, 2011
- 22 Folkens Dave, 5 Tips: Content Sharing Beyond Facebook [online];
21 January, 2011
URL: <http://www.toprankblog.com/2011/01/content-sharing-beyond-facebook/>
Accessed: 11 October, 2011

- 23 HP EliteBook 8530w [online], HP EliteBook 8530w Mobile Workstation – specifications and warranty
URL: <http://h10010.www1.hp.com/wwpc/ca/en/sm/WF06a/321957-321957-64295-3955549-3955549-3781677.html>
Accessed: 12 November, 2011
- 24 Zyxel [online], P-660HW Series
URL: http://www.zyxel.com/uk/en/products_services/p_660hw_series.shtml
Accessed: 12 November, 2011
- 25 Before you test your speed, Speedtest [online]
URL: <http://speedtest.net/>
Accessed: 12 November, 2011